Java

What is Java? / Why Java?

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. The idea behind Java is *Write Once, Run Anywhere*

What is JRE / JDK / JVM?

Java Virtual Machine

The Java Virtual Machine (JVM) is the virtual machine that runs the Java bytecodes. The JVM doesn't understand Java source code; that's why you compile your *.java files to obtain *.class files that contain the bytecodes understood by the JVM. It's also the entity that allows Java to be a "portable language" (write once, run anywhere). Indeed, there are specific implementations of the JVM for different systems (Windows, Linux, macOS, see the Wikipedia list), the aim is that with the same bytecodes they all give the same results.

Java Runtime Environment (JRE)

The Java Runtime Environment (JRE) provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language. In addition, two key deployment technologies are part of the JRE: Java Plug-in, which enables applets to run in popular browsers; and Java Web Start, which deploys standalone applications over a network. It is also the foundation for the technologies in the Java 2 Platform, Enterprise Edition (J2EE) for enterprise software development and deployment. The JRE does not contain tools and utilities such as compilers or debuggers for developing applets and applications.

Java Development Kit (JDK)

The JDK is a superset of the JRE, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications.

What is an object / class?

Object

A Java object is a combination of data and procedures working on the available data. An object has a state and behavior. The state of an object is stored in fields (variables), while methods (functions) display the object's behavior. Objects are created from templates known as classes.

Class

A class, in the context of Java, are templates that are used to create objects, and to define object data types and methods. Core properties include the data types and methods that may be used by the object. All class objects should have the basic class properties. Classes are categories, and objects are items within each category.

What is the root class from which every class extends?

java.lang.Object class is superclass of all classes. Class Object is the root of the class hierarchy. Every class has Object as a superclass. All objects, including arrays, implement the methods of this class.

What are the 4 pillars of OOP / Explain each (and how you've used them in your projects) ?

Abstraction

Abstraction is a process of exposing essential feature of an entity while hiding other irrelevant detail. Why would you want to use abstraction?

Abstraction reduces code complexity and at the same time it makes it aesthetically pleasant.

Encapsulation

We have to take in consideration that Encapsulation is somehow related to Data Hiding. Encapsulation is when you hide your modules internal data and all other implementation details/mechanism from other modules. it is also a way of restricting access to certain properties or component. *(Think Access Modifiers)*

Remember, Encapsulation is not data hiding, but Encapsulation leads to data hiding.

Inheritance

Like the word Inheritance literally means it is a practice of passing on property, titles, debts, rights and obligations upon the death of an individual. in OOP this is somehow true (Except the death of an individual), where the base class (the existing class sometimes called as the Parent class) has properties and methods that will be inherited by the sub class (sometimes called a subtype or child class) and it can have additional properties or methods.

The ability of creating a new class from an existing class.

Polymorphism

Just like in biology, Polymorphism refers to the ability to take into different forms or stages. A subclass can define its own unique behavior and still share the same functionalities or behavior of its parent/base class. Subclasses can have their own behavior and share some of its behavior from its parent class not the other way around. A parent class cannot have the behavior of its subclass.

Polymorphism is the ability of an object to change behavior on runtime

What is the difference between an abstract class and an interface?

Abstract Class

Abstract class in Java is similar to interface except that it can contain default method implementation. An abstract class can have an abstract method without body and it can have methods with implementation also.

The abstract keyword is used to create an abstract class and method. Abstract class in java can't be instantiated. An abstract class is mostly used to provide a base for subclasses to extend and implement the abstract methods and override or use the implemented methods in abstract class.

Like a class, an interface can have methods and variables, but the methods declared in interface are by default abstract (only method signature, no body).

Interfaces:

Interfaces specify what a class must do and not how. It is the blueprint of the class.

· An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move(). So, it specifies a set of methods that the class has to implement.

· If a class implements an interface and does not provide method bodies for all functions specified in the interface, then class must be declared abstract.

· A Java library example is, Comparator Interface. If a class implements this interface, then it can be used to sort a collection.

What are the implicit modifiers for interface variables?

Interfaces declared directly within a namespace can be declared as public or internal and, just like classes and structs, interfaces default to internal access. Interface members are always public because the purpose of an interface is to enable other types to access a class or struct. No access modifiers can be applied to interface members.

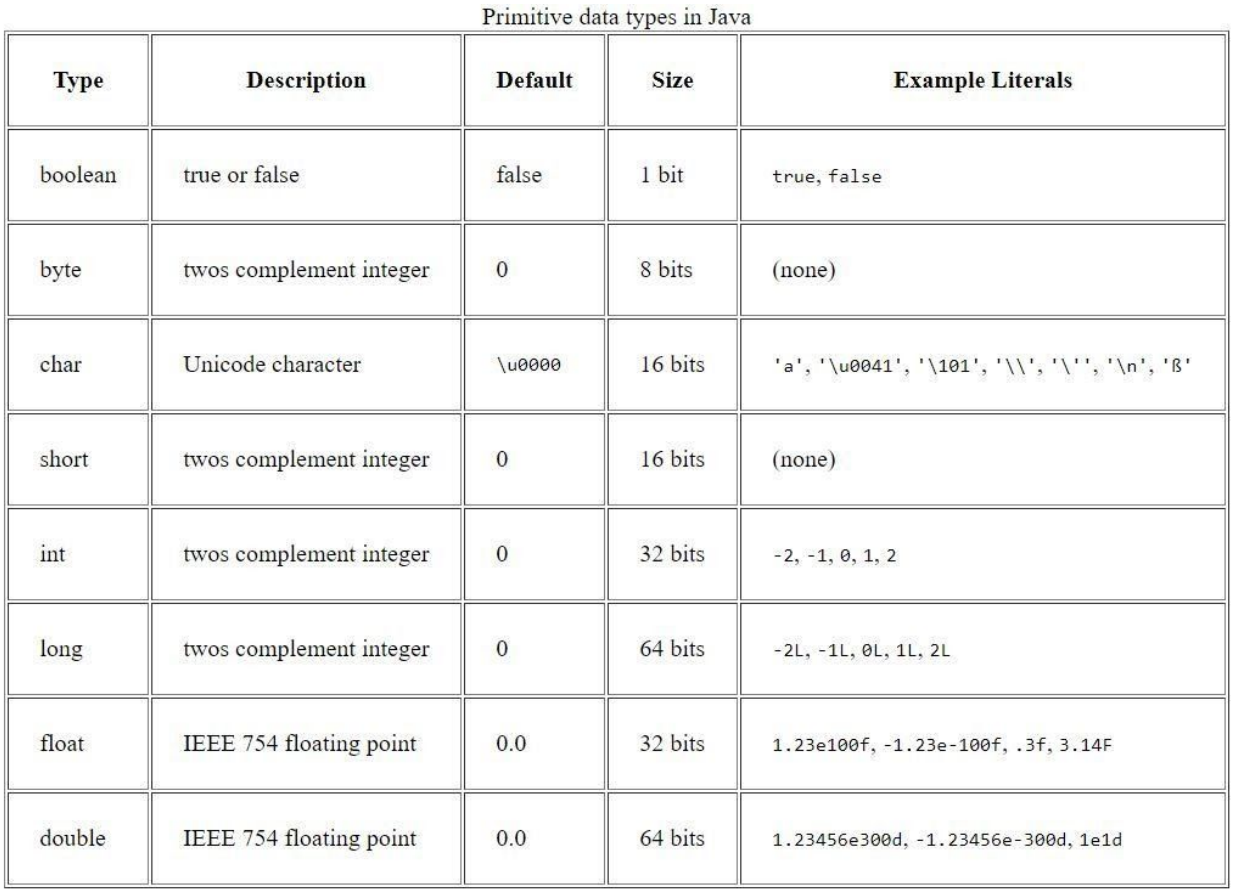
Can we use Access Modifier with Interface?
Yes, they can be declared as public or internal

By default, are interfaces internal?
Yes.

What about Access Modifiers with Interface members?
They are public. No access modifiers can be applied to interface members

.

What are the primitive data types in Java?



Primitive data types in Java

Type	Description	Default	Size	Example Literals
boolean	true or false	false	1 bit	`true`, `false`
byte	twos complement integer	0	8 bits	(none)
char	Unicode character	`\u0000`	16 bits	`'a'`, `'\u0041'`, `'\101'`, `'\\'`, `'\''`, `'\n'`, `'ß'`
short	twos complement integer	0	16 bits	(none)
int	twos complement integer	0	32 bits	`-2`, `-1`, `0`, `1`, `2`
long	twos complement integer	0	64 bits	`-2L`, `-1L`, `0L`, `1L`, `2L`
float	IEEE 754 floating point	0.0	32 bits	`1.23e100f`, `-1.23e-100f`, `.3f`, `3.14F`
double	IEEE 754 floating point	0.0	64 bits	`1.23456e300d`, `-1.23456e-300d`, `1e1d`

What is the difference between method overloading and overriding?

Method Overloading

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

Method Overriding

In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its superclasses or parent classes. When a method in a subclass has the same name, same parameters or signature and same return type (or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

Difference between extends and implements?

Extends

Extends is the keyword used to inherit the properties of a class.

Implements

Implements means you are using the elements of a Java Interface in your class. *Therefore, extends means that you are creating a subclass of the base class you are extending. You can only extend one class in your child class, but you can implement as many interfaces as you would like.*

What are collections in Java?

A Collection is a group of individual objects represented as a single unit. Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.

The Collection interface (java.util.Collection) and Map interface (java.util.Map) are the two main “root” interfaces of Java collection classes.

What are the interfaces in the Collections API?

The **Collection** in Java is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

What is the difference between a Set and a List?

Set

A Set is a Collection that cannot contain duplicate elements. It models the mathematical set abstraction. The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited. Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ. Two Set instances are equal if they contain the same elements.

The Java platform contains three general-purpose Set implementations: HashSet, TreeSet, and LinkedHashSet. HashSet, which stores its elements in a hash table, is the best-performing implementation; however, it makes no guarantees concerning the order of iteration. TreeSet, which stores its elements in a red-black tree, orders its elements based on their values; it is substantially slower than HashSet. LinkedHashSet, which is implemented as a hash table with a linked list running through it, orders its elements based on the order in which they were inserted into the set (insertion-order). LinkedHashSet spares its clients from the unspecified, generally chaotic ordering provided by HashSet at a cost that is only slightly higher.

List

The Java.util.List is a child interface of Collection. It is an ordered collection of objects in which duplicate values can be stored. Since List preserves the insertion order, it allows positional access and insertion of elements. List Interface is implemented by ArrayList, LinkedList, Vector and Stack classes.

What is the difference between an Array and an ArrayList?

Array

An array, in the context of Java, is a dynamically-created object that serves as a container to hold constant number of values of the same type. By declaring an array, memory space is allocated for values of a particular type. At the time of creation, the length of the array must be specified and remains constant.

To access an array element, the numerical index (a non-negative value) corresponding to the location of that element must be used. The first index value in the array is zero, thus, the index with value four is used to access the fifth element in the array. An array element that is also an array is known as a subarray. Arrays could have one or two dimensions.

ArrayList

The ArrayList class is a resizable array, which can be found in the java.util package. *The difference between a built-in array and an ArrayList in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one). While elements can be added and removed from an ArrayList whenever you want.*

What is the difference between ArrayList and Vector?

Vector

The Vector class implements a growable array of objects. Vectors basically falls in legacy classes but now it is fully compatible with collections.

· Vector implements a dynamic array that means it can grow or shrink as required. Like an array, it contains components that can be accessed using an integer index

· They are very similar to ArrayList but Vector is synchronised and have some legacy method which collection framework does not contain.

· It extends AbstractList and implements List interfaces.

Vector is similar to ArrayList, but it is synchronized. ArrayList is a better choice if your program is thread-safe. Vector and ArrayList require more space as more elements are added. Vector each time doubles its array size, while ArrayList grow 50% of its size each time.

What is the difference between TreeSet and HashSet?

Tree Set

TreeSet is one of the most important implementations of the SortedSet interface in Java that uses a Tree for storage. The ordering of the elements is maintained by a set using their natural ordering whether or not an explicit comparator is provided. This must be consistent with equals if it is to correctly implement the Set interface. It can also be ordered by a Comparator provided at set creation time, depending on which constructor is used. The TreeSet implements a NavigableSet interface by inheriting AbstractSet class.

Important features of TreeSet are as follows:

· TreeSet implements the SortedSet interface so duplicate values are not allowed.

· Objects in a TreeSet are stored in a sorted and ascending order.

· TreeSet does not preserve the insertion order of elements but elements are sorted by keys.

· TreeSet does not allow to insert Heterogeneous objects. It will throw classCastException at Runtime if trying to add heterogeneous objects.

· TreeSet serves as an excellent choice for storing large amounts of sorted information which are supposed to be accessed quickly because of its faster access and retrieval time.
· TreeSet is basically implementation of a self-balancing binary search tree like Red-Black Tree. Therefore operations like add, remove and search take O(Log n) time. And operations like printing n elements in sorted order takes O(n) time.

<u>Hash Set</u>

The HashSet class implements the Set interface, backed by a hash table which is actually a HashMap instance. No guarantee is made as to the iteration order of the set which means that the class does not guarantee the constant order of elements over time. This class permits the null element. The class also offers constant time performance for the basic operations like add, remove, contains and size assuming the hash function disperses the elements properly among the buckets, which we shall see further in the article.

important features of HashSet are:

· Implements Set Interface.
· Underlying data structure for HashSet is hashtable.
· As it implements the Set Interface, duplicate values are not allowed.
· Objects that you insert in HashSet are not guaranteed to be inserted in same order. Objects are inserted based on their hash code.
· NULL elements are allowed in HashSet.
· HashSet also implements Searlizable and Cloneable interfaces.

First major difference between HashSet and TreeSet is performance. HashSet is faster than TreeSet and should be preferred choice if sorting of element is not required. HashSet doesn't guaranteed any order while TreeSet maintains objects in Sorted order defined by either Comparable or Comparator method in Java.

What is the difference between HashTable and HashMap?

<u>HashTable</u>

This class implements a hash table, which maps keys to values. Any non-null object can be used as a key or as a value.
To successfully store and retrieve objects from a hashtable, the objects used as keys must implement the hashCode method and the equals method.

· It is similar to HashMap, but is synchronized.
· Hashtable stores key/value pair in hash table.

· In Hashtable we specify an object that is used as a key, and the value we want to associate to that key. The key is then hashed, and the resulting hash code is used as the index at which the value is stored within the table.
Constructors:

· Hashtable(): This is the default constructor.
· Hashtable(int size): This creates a hash table that has initial size specified by size.
· Hashtable(int size, float fillRatio): This version creates a hash table that has initial size specified by size and fill ratio specified by fillRatio. fill ratio: Basically, it determines how full hash table can be before it is resized upward. And its Value lie between 0.0 to 1.0
· Hashtable(Map m): This creates a hash table that is initialized with the elements in m.

HashMap

HashMap is a part of Java's collection since Java 1.2. It provides the basic implementation of Map interface of Java. It stores the data in (Key, Value) pairs. To access a value, one must know its key. HashMap is known as HashMap because it uses a technique called Hashing. Hashing is a technique of converting a large String to small String that represents same String. A shorter value helps in indexing and faster searches. HashSet also uses HashMap internally. It internally uses a link list to store key-value pairs already explained in HashSet in detail and further articles.

· HashMap is a part of java.util package.
· HashMap extends an abstract class AbstractMap which also provides an incomplete implementation of Map interface.
· It also implements Cloneable and Serializable interface. K and V in the above definition represent Key and Value respectively.
· HashMap doesn't allow duplicate keys but allows duplicate values. That means A single key can't contain more than 1 value but more than 1 key can contain a single value.
· HashMap allows null key also but only once and multiple null values.
· This class makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over time. It is roughly similar to HashTable but is unsynchronized.
· Are Maps in the Collections API?
· No maps are not in the collections API

HashMap allows null values as key and value whereas Hashtable doesn't allow nulls). HashMap does not guarantee that the order of the map will remain constant over time. HashMap is non-synchronized whereas Hashtable is synchronized.

Where are Strings stored?

We all know that JVM divides the allocated memory to a Java program into two parts. one is **Stack** and another one is **Heap**. Stack is used for execution purpose and heap is used for storage purpose. In that heap memory, JVM allocates some memory specially meant for string literals. This part of the heap memory is called String Constant Pool.
Whenever you create a string object using string literal, that object is stored in the string constant pool and whenever you create a string object using new keyword, such object is stored in the heap memory.

What are generics? What is the diamond operator (<>)?

<u>Generics</u>

Generics allow type (Integer, String, … etc. + user defined types) to be a parameter to methods, classes and interfaces. For example, classes like HashSet, ArrayList, HashMap, etc. use generics very well. We can use them for any type.

<u>Diamond Operator <></u>

The Purpose of diamond operator is to simplify the use of generics when creating an object. It avoids unchecked warnings in a program as well as reducing generic verbosity by not requiring explicit duplicate specification of parameter types.

What are enumerations (enums)?

A Java Enum is a special Java type used to define collections of constants. More precisely, a Java enum type is a special kind of Java class. An enum can contain constants, methods etc.

What are annotations?

Annotations are used to provide supplement information about a program.
· Annotations start with ‘@’.
· Annotations do not change action of a compiled program.
· Annotations help to associate metadata (information) to the program elements i.e. instance variables, constructors, methods, classes, etc.
· Annotations are not pure comments as they can change the way a program is treated by compiler.

There are 3 categories of Annotations:

1. Marker Annotations:
The only purpose is to mark a declaration. These annotations contain no members and do not consist any data. Thus, its presence as an annotation is sufficient. Since, marker interface contains no members, simply determining whether it is present or absent is sufficient. @Override is an example of Marker Annotation.

Example: - @TestAnnotation()

2. Single value Annotations:

These annotations contain only one member and allow a shorthand form of specifying the value of the member. We only need to specify the value for that member when the annotation is applied and don’t need to specify the name of the member. However, in order to use this shorthand, the name of the member must be value.

Example: - @TestAnnotation(“testing”);

3. Full Annotations:

These annotations consist of multiple data members/ name, value, pairs.

Example:- @TestAnnotation(owner=”Rahul”, value=”Class Geeks”)

Explain stack vs heap?

Whenever an object is created, it's always stored in the **Heap** space and **stack memory** contains the reference to it. Stack memory only contains local primitive variables and reference variables to objects in heap space. Stack memory size is very low when compared to Heap memory.

How do you serialize / deserialize an object in Java?

Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.

The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.

To make a Java object serializable we implement the java.io.Serializable interface.
The ObjectOutputStream class contains writeObject() method for serializing an Object.

```
public final void writeObject(Object obj)
                   throws IOException
```

The ObjectInputStream class contains readObject() method for deserializing an object.

```
public final Object readObject()
               throws IOException,
            ClassNotFoundException
```

Advantages of Serialization:
1. To save/persist state of an object.
2. To travel an object across a network.
Only the objects of those classes can be serialized which are implementing javaOnly the objects of those classes can be serialized which are implementing java.io.Serializable interface.
Serializable is a marker interface (has no data member and method). It is used to “mark” java classes so that objects of these classes may get certain capability. Other examples of marker interfaces are:- Cloneable and Remote.

Points to Remember:
1. If a parent class has implemented Serializable interface then child class doesn’t need to implement it but vice-versa is not true.
2. Only non-static data members are saved via Serialization process.
3. Static data members and transient data members are not saved via Serialization process. So, if you don’t want to save value of a non-static data member then make it transient.
4. Constructor of object is never called when an object is deserialized.
5. Associated objects must be implementing Serializable interface..io.Serializable interface.

Serializable is a marker interface (has no data member and method). It is used to “mark” java classes so that objects of these classes may get certain capability. Other examples of marker interfaces are:- Cloneable and Remote.

Points to remember
1. If a parent class has implemented Serializable interface then child class doesn’t need to implement it but vice-versa is not true.
2. Only non-static data members are saved via Serialization process.
3. Static data members and transient data members are not saved via Serialization process. So, if you don’t want to save value of a non-static data member then make it transient.
4. Constructor of object is never called when an object is deserialized.
5. Associated objects must be implementing Serializable interface.

What is a POJO? What is a bean?

POJO stands for Plain Old Java Object. It is an ordinary Java object, not bound by any special restriction other than those forced by the Java Language Specification and not requiring any class path. POJOs are used for increasing the readability and re-usability of a program. POJOs have gained most acceptance because they are easy to write and understand.

Beans are special type of Pojos. There are some restrictions on POJO to be a bean.

· All JavaBeans are POJOs but not all POJOs are JavaBeans.
· Serializable i.e. they should implement Serializable interface. Still some POJOs who don’t implement Serializable interface are called POJOs because Serializable is a marker interface and therefore not of much burden.
· Fields should be private. This is to provide the complete control on fields.
· Fields should have getters or setters or both.
· A no-arg constructor should be there in a bean.
· Fields are accessed only by constructor or getter setters.

POJO classes and Beans both are used to define java objects to increase their readability and reusability. POJOs don’t have other restrictions while beans are special POJOs with some restrictions.

What is the difference between final, .finalize(), and finally?

final: final keyword can be used for class, method and variables. A final class cannot be subclassed and it prevents other programmers from subclassing a secure class to invoke insecure methods. A final method can't be overridden. A final variable can't change from its initialized value.

finalize(): finalize method is used just before an object is destroyed and called just prior to garbage collection.

finally: finally, a key word used in exception handling, creates a block of code that will be executed after a try/catch block has completed and before the code following the try/catch block. The finally block will execute whether or not an exception is thrown. For example, if a method opens a file upon exit, then you will not want the code that closes the file to be bypassed by the exception-handling mechanism. This finally keyword is designed to address this contingency.

Do you need a catch block? Can have more than 1? Order of them?

Java **try block** is used to enclose the code that might throw an exception. It must be used within the method.

If an exception occurs at the particular statement of try block, the rest of the block code will not execute. So, it is recommended not to keeping the code in try block that will not throw an exception.

Java try block must be followed by either catch or finally block.

Java **catch block** is used to handle the Exception by declaring the type of exception within the parameter. The declared exception must be the parent class exception (i.e., Exception) or the generated exception type. However, the good approach is to declare the generated type of exception.

The catch block must be used after the try block only. You can use multiple catch block with a single try block.

What is base class of all exceptions? What interface do they all implement?

<u>Throwable</u>

The java.lang.Throwable class is the superclass of all errors and exceptions in the Java language. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java throw statement

List some checked and unchecked exceptions?

Checked exceptions

A checked exception is an exception that is checked (notified) by the compiler at compilation-time, these are also called as compile time exceptions. These exceptions cannot simply be ignored, the programmer should take care of (handle) these exceptions.

- IOException
- SQLException
- DataAccessException
- ClassNotFoundException
- InvocationTargetException
- MalformedURLException

Unchecked exceptions

An unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions. These include programming bugs, such as logic errors or improper use of an API. Runtime exceptions are ignored at the time of compilation.

- NullPointerException
- ArrayIndexOutOfBoundsException
- ArithmeticException
- IllegalArgumentException
- NumberFormatException

What is a Marker interface?

The marker interface pattern is a design pattern in computer science, used with languages that provide run-time type information about objects. It provides a means to associate metadata with a class where the language does not have explicit support for such metadata.

It is an empty interface (no field or methods). Examples of marker interface are Serializable, Cloneable and Remote interface. All these interfaces are empty interfaces.
Examples of Marker Interface which are used in real-time applications :

Cloneable interface:

Cloneable interface is present in java.lang package. There is a method clone() in Object class. A class that implements the Cloneable interface indicates that it is legal for clone() method to make a field-for-field copy of instances of that class.

Invoking Object's clone method on an instance of the class that does not implement the Cloneable interface results in an exception CloneNotSupportedException being thrown. By convention, classes that implement this interface should override Object.clone() method.

Serializable interface

: Serializable interface is present in java.io package. It is used to make an object eligible for saving its state into a file. This is called Serialization.
Classes that do not implement this interface will not have any of their state serialized or deserialized. All subtypes of a serializable class are themselves serializable.

Remote interface:

Remote interface is present in java.rmi package. A remote object is an object which is stored at one machine and accessed from another machine. So, to make an object a remote object, we need to flag it with Remote interface. Here, Remote interface serves to identify interfaces whose methods may be invoked from a non-local virtual machine. Any object that is a remote object must directly or indirectly implement this interface. RMI (Remote Method Invocation) provides some convenience classes that remote object implementations can extend which facilitate remote object creation.

Why are strings immutable in java?

The string is Immutable in Java because String objects are cached in String pool. Since cached String literals are shared between multiple clients there is always a risk, where one client's action would affect all another client. For example, if one client changes the value of String "Test" to "TEST", all other clients will also see that value as explained in the first example. Since caching of String objects was important from performance reason this risk was avoided by making String class Immutable.

What is the difference between String, StringBuilder, and StringBuffer?

String Builder is Mutable and not threadsafe

String Buffer is threadsafe

What are the access modifiers in Java? Explain what they are and how you use them.

Public

The Java access modifier public means that all code can access the class, field, constructor or method, regardless of where the accessing code is located. The accessing code can be in a different class and different package.

Private

If a method or variable is marked as private (has the private access modifier assigned to it), then only code inside the same class can access the variable, or call the method. Code inside subclasses cannot access the variable or method, nor can code from any external class. Classes cannot be marked with the private access modifier. Marking a class with the private access modifier would mean that no other class could access it, which means that you could not really use the class at all. Therefore, the private access modifier is not allowed for classes.

Protected

The protected access modifier provides the same access as the default access modifier, with the addition that subclasses can access protected methods and member variables (fields) of the superclass. This is true even if the subclass is not located in the same package as the superclass.

Default

The default Java access modifier is declared by not writing any access modifier at all. The default access modifier means that code inside the class itself as well as code inside classes in the same package as this class, can access the class, field, constructor or method which the default access modifier is assigned to. Therefore, the default access modifier is also sometimes referred to as the package access modifier.

What are the non-access modifiers in Java?

Java provides a number of non-access modifiers to achieve many other functionalities.

- The **static** modifier for creating class methods and variables.
- The **final** modifier for finalizing the implementations of classes, methods, and variables.
- The **abstract** modifier for creating abstract classes and methods.
- The **synchronized** and volatile modifiers, which are used for threads.

Static

Static Variables

The static keyword is used to create variables that will exist independently of any instances created for the class. Only one copy of the static variable exists regardless of the number of instances of the class.
Static variables are also known as class variables. Local variables cannot be declared static.

Static Methods

The static keyword is used to create methods that will exist independently of any instances created for the class.
Static methods do not use any instance variables of any object of the class they are defined in. Static methods take all the data from parameters and compute something from those parameters, with no reference to variables.
Class variables and methods can be accessed using the class name followed by a dot and the name of the variable or method.

Final

Final Variables

A final variable can be explicitly initialized only once. A reference variable declared final can never be reassigned to refer to a different object.
However, the data within the object can be changed. So, the state of the object can be changed but not the reference.
With variables, the final modifier often is used with static to make the constant a class variable.

Final Methods

A final method cannot be overridden by any subclasses. As mentioned previously, the final modifier prevents a method from being modified in a subclass.

The main intention of making a method final would be that the content of the method should not be changed by any outsider.

Final Classes

The main purpose of using a class being declared as final is to prevent the class from being subclassed. If a class is marked as final then no class can inherit any feature from the final class.

Abstract

Abstract Class

An abstract class can never be instantiated. If a class is declared as abstract then the sole purpose is for the class to be extended.

A class cannot be both abstract and final (since a final class cannot be extended). If a class contains abstract methods then the class should be declared abstract. Otherwise, a compile error will be thrown.

An abstract class may contain both abstract methods as well normal methods.

Abstract Methods

An abstract method is a method declared without any implementation. The methods body (implementation) is provided by the subclass. Abstract methods can never be final or strict. Any class that extends an abstract class must implement all the abstract methods of the super class, unless the subclass is also an abstract class.

If a class contains one or more abstract methods, then the class must be declared abstract. An abstract class does not need to contain abstract methods.

The Synchronized Modifier

The synchronized keyword used to indicate that a method can be accessed by only one thread at a time. The synchronized modifier can be applied with any of the four access level modifiers.

The Transient Modifier

An instance variable is marked transient to indicate the JVM to skip the particular variable when serializing the object containing it.

This modifier is included in the statement that creates the variable, preceding the class or data type of the variable.

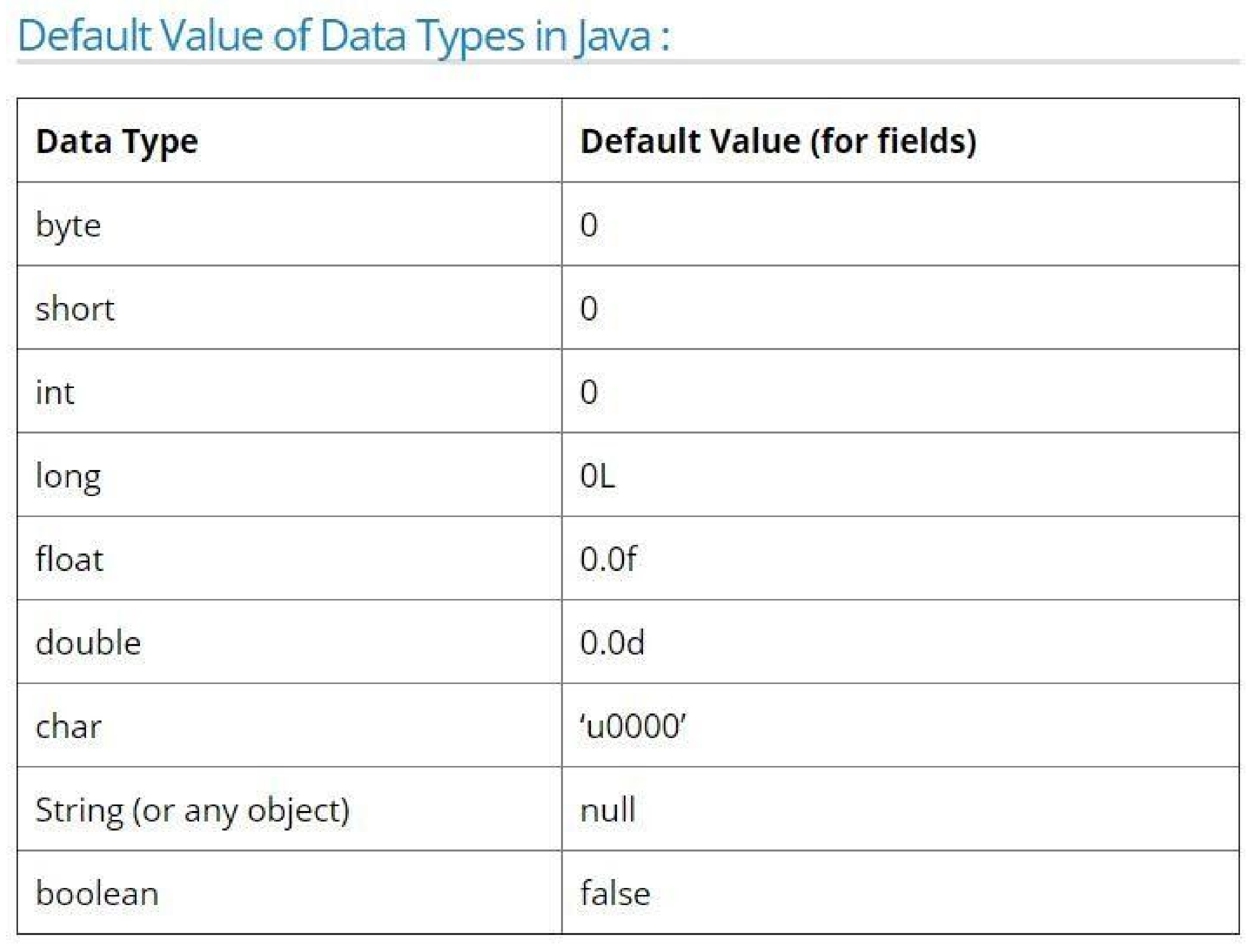
The Volatile Modifier

The volatile modifier is used to let the JVM know that a thread accessing the variable must always merge its own private copy of the variable with the master copy in the memory. Accessing a volatile variable synchronizes all the cached copied of the variables in the main memory. Volatile can only be applied to instance variables, which are of type object or private. A volatile object reference can be null.

What is the difference between static and final variables?

The static member can be accessed before the class object is created. Final keyword is used to declare, a constant variable, a method which cannot be overridden and a class that cannot be inherited.

What are the default values for all data types in Java?



Default Value of Data Types in Java :

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	‘u0000’
String (or any object)	null
boolean	false

What are the implicit modifiers for interface variables / methods?

Static & Final by default

What are transient variables?

transient is a Java keyword which marks a member variable not to be serialized when it is persisted to streams of bytes. When an object is transferred through the network, the object needs to be 'serialized'. Serialization converts the object state to serial bytes.

What is a wrapper class?

A Wrapper class is a class whose object wraps or contains a primitive data type. When we create an object to a wrapper class, it contains a field and, in this field, we can store a primitive data type. In other words, we can wrap a primitive value into a wrapper class object.

- They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).
- The classes in java.util package handles only objects and hence wrapper classes help in this case also.
- Data structures in the Collection framework, such as ArrayList and Vector, store only objects (reference types) and not primitive types.
- An object is needed to support synchronization in multithreading.

What is autoboxing / unboxing?

Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this is called **unboxing.**

What is Reflection API?

Reflection is an API which is used to examine or modify the behavior of methods, classes & interfaces at runtime. Reflection gives us information about the class to which an object belongs and also the methods of that class which can be executed by using the object.

Is Java pass-by-value or pass-by-reference?

Java is Pass by Value and Not Pass by Reference. The method parameter values are copied to another variable and then the copied object is passed, that's why it's called pass by value.

What is the difference between == and .equals()?

In general both equals() and "==" operator in Java are used to compare objects to check equality but here are some of the differences between the two:

- Main difference between .equals() method and == operator is that one is method and other is operator.

- We can use == operators for reference comparison (address comparison) and .equals() method for content comparison. In simple words, == checks if both objects point to the same memory location whereas .equals() evaluates to the comparison of values in the objects.

- If a class does not override the equals method, then by default it uses equals(Object o) method of the closest parent class that has overridden this method. See this for detail

What are functional interfaces?

A functional interface is an interface that contains only one abstract method. They can have only one functionality to exhibit. From Java 8 onwards, lambda expressions can be used to represent the instance of a functional interface.

What are lambdas?

Lambda expressions basically express instances of functional interfaces (An interface with single abstract method is called functional interface. An example is java.lang.Runnable). lambda expressions implement the only abstract function and therefore implement functional interfaces

Lambda expressions are added in Java 8 and provide below functionalities.

- Enable to treat functionality as a method argument, or code as data.
- A function that can be created without belonging to any class.
- A lambda expression can be passed around as if it was an object and executed on demand.

Throw vs. Throws vs. Throwable

Throw:

The throw keyword in Java is used to explicitly throw an exception from a method or any block of code. We can throw either checked or unchecked exception. The throw keyword is mainly used to throw custom exceptions.

Throws

throws is a keyword in Java which is used in the signature of method to indicate that this method might throw one of the listed type exceptions. The caller to these methods has to handle the exception using a try-catch block.

Throwable

The java.lang.Throwable class is the superclass of all errors and exceptions in the Java language. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java throw statement.

First line of constructor?

Each time an object is created using new() keyword at least one constructor (it could be default constructor) is invoked to assign initial values to the data members of the same class.

```
class Geek
{
  .......


  // A Constructor
  new Geek() {}
```

What is JUnit?

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development.

What is TDD?

Test-driven development is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests

What are the annotations in JUnit? Order of execution?

@BeforeClass – Run once before any of the test methods in the class, public static void
@AfterClass – Run once after all the tests in the class have been run, public static void
@Before – Run before @Test, public void
@After – Run after @Test, public void
@Test – This is the test method to run, public void

By default, JUnit runs tests using a deterministic, but unpredictable order. In most cases, that behavior is perfectly fine and acceptable; but there are cases when we need to enforce a specific ordering.

Give an example of a test case?

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.Test;
public class MyTests {
    @Test
    public void multiplicationOfZeroIntegersShouldReturnZero() {
        MyClass tester = new MyClass(); // MyClass is tested
        // assert statements
        assertEquals(0, tester.multiply(10, 0), "10 x 0 must be
0");
        assertEquals(0, tester.multiply(0, 10), "0 x 10 must be
0");
        assertEquals(0, tester.multiply(0, 0), "0 x 0 must be
0");
    }
}
```

What is Maven?

Maven is a build automation tool used primarily for Java projects. Maven addresses two aspects of building software: first, it describes how software is built, and second, it describes its dependencies.

What is the Maven lifecycle?

1. **validate** - validate the project is correct and all necessary information is available
2. **compile** - compile the source code of the project
3. **test** - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
4. **package -** take the compiled code and package it in its distributable format, such as a JAR.
5. **verify -** run any checks on results of integration tests to ensure quality criteria are met
6. **install -** install the package into the local repository, for use as a dependency in other projects locally
7. **deploy -** done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

Where / when does Maven retrieve dependencies from? Where are they stored locally?

A repository in Maven holds build artifacts and dependencies of varying types.

There are exactly two types of repositories: local and remote. The local repository is a directory on the computer where Maven runs. It caches remote downloads and contains temporary build artifacts that you have not yet released.

What is the POM and what is the pom.xml?

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.

What methods are available in the Object class?

java.lang.Object provides a number of methods that are common to all objects. toString() is the most common such method. Since the default toString() method only produces the name of the class, you should override it in all classes you define.

- public Object()
- public final Class getClass()
- public int hashCode()
- public boolean equals(Object obj)
- protected Object clone() throws CloneNotSupportedException
- public String toString()
- public final void notify()
- public final void notifyAll()
- public final void wait(long timeout) throws InterruptedException
- public final void wait(long timeout, int nanoseconds) throws InterruptedException
- public final void wait() throws InterruptedException
- protected void finalize() throws Throwable

How would you clone an object?

The object cloning is a way to create exact copy of an object. The clone() method of Object class is used to clone an object. The java.lang.Cloneable interface must be implemented by the class whose object clone we want to create.

What is an enhanced for loop and what is a forEach loop?

The enhanced for loop was introduced in Java 5 as a simpler way to iterate through all the elements of a Collection. It can also be used for arrays, but this is not the original intended purpose.

What is try-with-resources? What interface must the resource implement to use this feature?

The try -with-resources statement is a try statement that declares one or more resources. A resource is an object that must be closed after the program is finished with it. Any object that implements **java.lang.AutoCloseable**, which includes all objects which implement **java.io.Closeable** , can be used as a resource.

What are covariant return types?

A covariant return type of a method is one that can be replaced by a "narrower" type when the method is overridden in a subclass.

What are 3 usages of super keyword?

1. to refer to immediate parent class instance variable.
2. super() is used to invoke immediate parent class constructor (also can pass params)
3. to invoke immediate parent class method.

Can you overload / override a main method? static method? a private method? a default method? a protected method?

You can overload the **main() method**, but only public static void main(String[] args) will be used when your class is launched by the JVM. You cannot override them.

It is ok to overload **static** (and **final**) methods. No,Static methods can't be overriden as it is part of a class rather than an object. But one can overload static method.

You can’t overload a **private** method. Constructors and **private** methods are not inherited, so they cannot be overridden.

A **default** method cannot override a method from java.lang.Object.

You cannot override a protected method.

Difference between FileReader and BufferedReader?

<u>FileReader</u>

Java FileReader Class. Java FileReader class is used to read data from the file. It returns data in byte format like FileInputStream class. It is character-oriented class which is used for file handling in java.

<u>BufferedReader</u>

Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

The buffer size may be specified, or the default size may be used. The default is large enough for most purposes.

In general, each read request made of a Reader causes a corresponding read request to be made of the underlying character or byte stream. It is therefore advisable to wrap a BufferedReader around any Reader whose read() operations may be costly, such as FileReaders and InputStreamReaders.

How to pass multiple values with a single parameter into a method?

Use varargs

What is static block?

Used for static initialization. Executed only once - upon creation of first object of class or access to static method of class

What is static imports?

Importing a static method or variable from a class - syntax: import static

How would you clone an object?

use the .clone() method inherited from Object class

What makes a class immutable?

Initialize all the fields via a constructor performing deep copy.

If two objects are equal, do they have the same hashcode? If not equal?

It is not required that if two objects are unequal according to the equals(java.lang.Object) method, then calling the hashCode method on each of the two objects must produce distinct integer results. You can have two different objects with the same hashcode.

What data types are supported in switch statements?

A switch works with the **byte , short , char , and int** primitive data types. It also works with enumerated types, the String class, and a few special classes that wrap certain primitive types: **Character , Byte , Short , and Integer**.

List all non-access modifiers?

Final, Abstract, Synchronized, Transient & Volatile

Which collections cannot hold null values?

HashSet, LinkedHashSet and TreeSet implementation do not allow null elements

If 2 interfaces have default methods and you implement both, what happens?

The code will NOT compile unless you override the method. However, the code WILL compile if one interface is implemented further up in the class hierarchy than the other - in this case, the closest method implementation in the hierarchy will be called

If 2 interfaces have same variable names and you implement both, what happens?

The code will compile unless you make a reference to the variable (this is an ambiguous reference). You must explicitly define the variable by using the interface name: int a = INTERFACENAME.a;

Why does HashTable not take null key?

The hash table hashes the keys given as input, and the null value cannot be hashed

Multi-catch block - can you catch more than one exception in a single catch block?

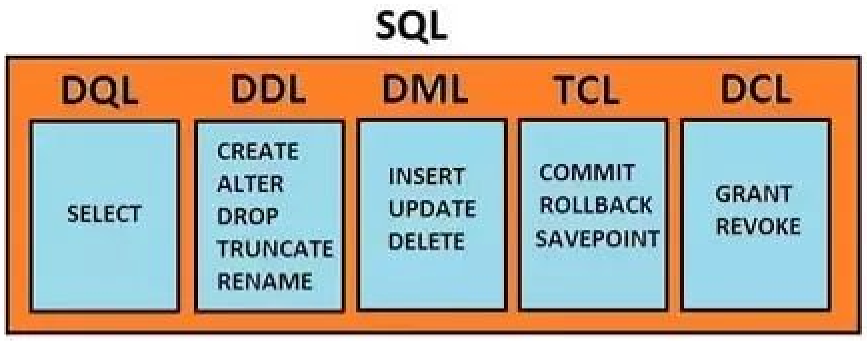
The catch block has been improved to handle multiple exceptions in a single catch block. If you are catching multiple exceptions and they have similar code, then using this feature will reduce code duplication.

SQL and JDBC

What is SQL?

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.

What are the sublanguages in SQL?



1.DDL (Data Definition Language)

These statements are used to create tables and databases and define field properties or table properties. Examples of commands that fall in this category are CREATE,ALTER and DROP statements

2. DML (Data Manipulation Language)

The statements that falls under this category are used to update data or add or remove data from tables. UPDATE, DELETE and

INSERT commands fall under this category.

3. DCL (Data Control Language)

It is used to control who access the data. The commands that come under this category are GRANT and REVOKE

4. TCL (Transaction Control Language)

This language is used to commit data and restore data. COMMIT and ROLLBACK falls under this category.

5. DQL (Data Query Language)

This is to retrieve data from sql server. SELECT statement falls in this category.

What commands do we see in these sublanguages?

See above.

What is a Primary Key?

A primary key is a special relational database table column (or combination of columns) designated to uniquely identify all table records. A primary key's main features are: It must contain a unique value for each row of data. It cannot contain null values.

What is a Composite Key?

In database design, a composite key is a candidate key that consists of two or more attributes that uniquely identify an entity occurrence. A compound key is a composite key for which each attribute that makes up the key is a simple key in its own right.

What is a Foreign Key?

In the context of relational databases, a foreign key is a field in one table that uniquely identifies a row of another table or the same table. In simpler words, the foreign key is defined in a second table, but it refers to the primary key or a unique key in the first table.

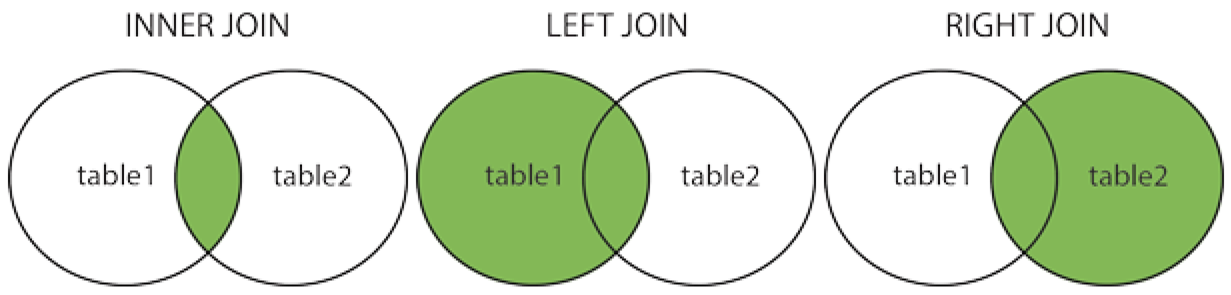
What are the different types of joins?

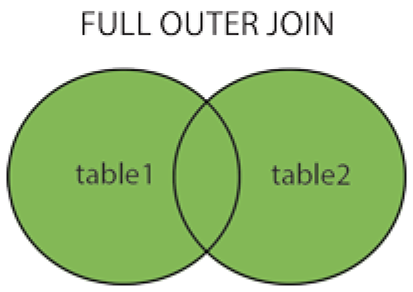
(INNER) JOIN: Returns records that have matching values in both tables

LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table

RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table

FULL (OUTER) JOIN: Return all records when there is a match in either left or right table





What are the different constraints in Oracle SQL?

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

NOT NULL - Ensures that a column cannot have a NULL value
UNIQUE - Ensures that all values in a column are different
PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
FOREIGN KEY - Uniquely identifies a row/record in another table
CHECK - Ensures that all values in a column satisfies a specific condition
DEFAULT - Sets a default value for a column when no value is specified
INDEX - Used to create and retrieve data from the database very quickly

Order By vs Group By?

ORDER BY alters the order in which items are returned. **GROUP BY** will aggregate records by the specified columns which allows you to perform aggregation functions on non-grouped columns (such as SUM, COUNT, AVG, etc). **ORDER BY: s**ort the data in ascending or descending order.

Why would we ever use a self join?

Self-joins are used to compare values in a column with other values in the same column in the same table. One practical use for self-joins: obtaining running counts and running totals in an SQL query.

What is referential Integrity?

Referential integrity (RI) is a relational database concept, which states that table relationships must always be consistent. In other words, any foreign key field must agree with the primary key that is referenced by the foreign key.

Explain Normalization.

Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

What is the difference between a join and a union?

Both joins and unions can be used to combine data from one or more tables into a single result.

Whereas a join is used to combine columns from different tables, the union is used to combine rows.

What is a view?

In a database, a view is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary.

What are nested queries and why do we use them?

Nested queries are queries that are embedded in the where clause of a query. We do this to specify a condition to further restrict retrieved data. (also known as subqueries)

What are indexes?

Indexes are a way to have fast access to data accessed frequently. They can be clustered and unclustered.

What are aggregate and scalar functions?

Aggregate functions are functions that operate on multiple values that return a single result. Scalar on the other hand operates on one value and returns a single value. Aggregate : Sum(), AVG(), COUNT(). Scalar - UCASE(), LCASE(), MID().

What are triggers and when can I set them to activate?

A trigger is a special type of stored procedure that automatically executes when an event occurs in the database server. DML triggers execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

What are cursors?

A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it. This temporary work area is used to store the data retrieved from the database, and manipulate this data.

Difference between implicit and explicit cursors?

Implicit Cursor

A SQL (implicit) cursor is opened by the database to process each SQL statement that is not associated with an explicit cursor. Every SQL (implicit) cursor has six attributes, each of which returns useful information about the execution of a data manipulation statement.

Explicit Cursor

An **explicit cursor** is defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row. We can provide a suitable name for the cursor.
General Syntax for creating a cursor is as given below:
CURSOR cursor_name IS select_statement;

- *cursor_name – A suitable name for the cursor.*
- *select_statement – A select query which returns multiple rows.*

How to use Explicit Cursor?

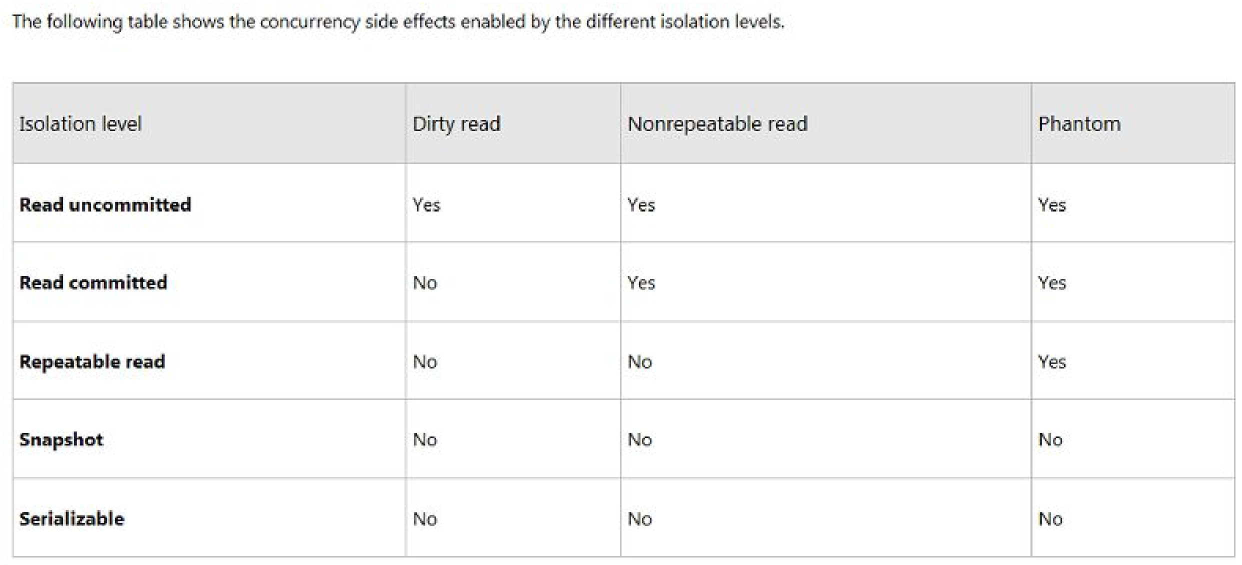
There are four steps in using an Explicit Cursor.

- DECLARE the cursor in the declaration section.
- OPEN the cursor in the Execution Section.

- FETCH the data from cursor into PL/SQL variables or records in the Execution Section.
- CLOSE the cursor in the Execution Section before you end the PL/SQL Block.

An implicit cursor is used for all SQL statements Declare, Open, Fetch, Close. An explicit cursors are used to process multirow SELECT statements An implicit cursor is used to process INSERT, UPDATE, DELETE and single row SELECT ____ INTO statements.

What are the transaction isolation levels and what anomalies do they protect against?



The following table shows the concurrency side effects enabled by the different isolation levels.

Isolation level	Dirty read	Nonrepeatable read	Phantom
Read uncommitted	Yes	Yes	Yes
Read committed	No	Yes	Yes
Repeatable read	No	No	Yes
Snapshot	No	No	No
Serializable	No	No	No

Explain multi-version concurrency control and what it is used for.

Multi-version concurrency control deals with the idea of multiple users keeping concurrent access to the database. This is kept with transaction isolation levels.

What is the difference between clustered and unclustered indexes?

An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view. An index contains keys built from one or more columns in the table or view. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently.
A table or view can contain the following types of indexes:

- **Clustered**

    - Clustered indexes sort and store the data rows in the table or view based on their key values. These are the columns included in the index definition. There can be only one clustered index per table, because the data rows themselves can be stored in only one order.
    - The only time the data rows in a table are stored in sorted order is when the table contains a clustered index. When a table has a clustered index, the table is called a clustered table. If a table has no clustered index, its data rows are stored in an unordered structure called a heap.
- **Nonclustered**
    - Nonclustered indexes have a structure separate from the data rows. A nonclustered index contains the nonclustered index key values and each key value entry has a pointer to the data row that contains the key value.
    - The pointer from an index row in a nonclustered index to a data row is called a row locator. The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table. For a heap, a row locator is a pointer to the row. For a clustered table, the row locator is the clustered index key.
    - You can add nonkey columns to the leaf level of the nonclustered index to by-pass existing index key limits, and execute fully covered, indexed, queries. For more information, see Create Indexes with Included Columns. For details about index key limits see Maximum Capacity Specifications for SQL Server.

Both clustered and nonclustered indexes can be unique. This means no two rows can have the same value for the index key. Otherwise, the index is not unique and multiple rows can share the same key value.

A clustered index spans across multiple columns while an unclustered index is for one column of data.

How do you add a column to a table?

You can use the ALTER TABLE statement in SQL Server to add a column to a table.

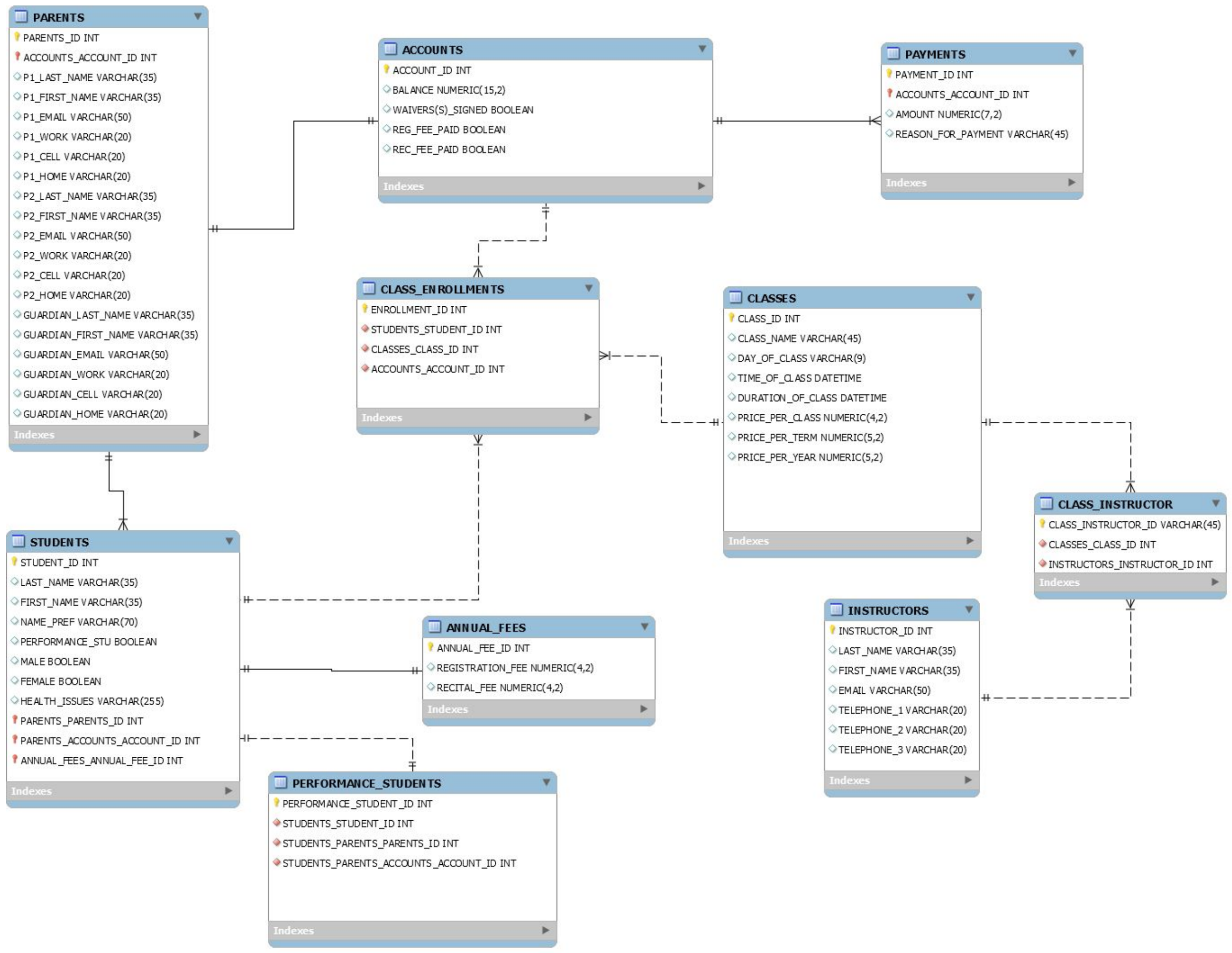
Syntax

The syntax to add a column in a table in SQL Server (Transact-SQL) is:

```
ALTER TABLE table_name
  ADD column_name column_definition;
```

What is an ERD?

An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and specifies relationships that can exist between entities.



What is the difference between WHERE and HAVING?

A WHERE clause is used is filter records from a result. The filter occurs before any groupings are made.

A HAVING clause is used to filter values from a group.

How do you prevent SQL injection in JDBC?

The problem with SQL injection is, that a user input is used as part of the SQL statement. By using prepared statements you can force the user input to be handled as the content of a parameter (and not as a part of the SQL command).

But if you don't use the user input as a parameter for your prepared statement but instead build your SQL command by joining strings together, you are still vulnerable to SQL injections even when using prepared statements.

What is the DAO pattern?

The Data Access Object (DAO) pattern is a structural pattern that allows us to **isolate the application/business layer from the persistence layer (usually a relational database, but it could be any other persistence mechanism) using an abstract API**.

Why create a DAO interface?

The functionality of this API is to hide from the application all the complexities involved in performing CRUD operations in the underlying storage mechanism. This permits both layers to evolve separately without knowing anything about each other.

T/F: The application should deal with the data alongside the DAO.

False, The Database Layer should be separate from the Logic Layer

What design pattern would we use if there are multiple DAO implementations?

Repository Pattern: In many applications, the business logic accesses data from data stores such as databases, SharePoint lists, or Web services. Directly accessing the data can result in the following: Duplicated code A higher potential for programming errors Weak typing of the business data Difficulty in centralizing data-related policies such as caching An inability to easily test the business logic in isolation from external dependencies

What is JDBC? What is its package name? What kind of exceptions does it throw?

Java Database Connectivity is an application programming interface for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation.

The JDBC API is comprised of two Java packages: **java.sql and javax.sql.**

Method	Description
getErrorCode( )	Gets the error number associated with the exception.
getMessage( )	Gets the JDBC driver's error message for an error, handled by the driver or gets the Oracle error number and message for a database error.
getSQLState( )	Gets the XOPEN SQLstate string. For a JDBC driver error, no useful information is returned from this method. For a database error, the five-digit XOPEN SQLstate code is returned. This method can return null.
getNextException( )	Gets the next Exception object in the exception chain.
printStackTrace( )	Prints the current exception, or throwable, and it's backtrace to a standard error stream.
printStackTrace(PrintStream s)	Prints this throwable and its backtrace to the print stream you specify.
printStackTrace(PrintWriter w)	Prints this throwable and it's backtrace to the print writer you specify.

What are the JDBC interfaces?

DriverManager

The DriverManager class (java.sql.DriverManager) is one of main components of JDBC. DriverManager manages database drivers, load database specific drivers and select the most appropriate database specific driver from the previously loaded drivers when a new connection is established. Managing, loading and selecting drivers are done automatically by the DriverManager from JDBC 4.0, when a new connection is created.

The DriverManager can also be considered as a connection factory class as it uses database specific drivers to create connection (java.sql.Connection) objects. DriverManager consist of only one private constructor and hence it cannot be inherited or initialized directly. All other members of DriverManager are static. DriverManager maintains a list of DriverInfo objects that hold one Driver object each.

Connection

Connection interface provides a standard abstraction to access the session established with a database server. JDBC driver provider should implement the connection interface. We can obtain a Connection object using the getConnection() method of the DriverManager as:

```
Connection con = DriverManager.getConnection(url, username,
password);
```

Before JDBC 4.0, we had to first load the Driver implementation before getting the connection from the DriverManager as:

```
Class.forName(oracle.jdbc.driver.OracleDriver);
```

We can also use DataSource for creating a connection for better data source portability and is the preferred way in production applications.

Statement

The Statement interface provides a standard abstraction to execute SQL statements and return the results using the ResultSet objects.

A Statement object contains a single ResultSet object at a time. The execute method of a Statement implicitly close its current ResultSet if it is already open. PreparedStatement and

CallableStatement are two specialized Statement interfaces. You can create a Statement object by using the createStatement() method of the Connection interface as:

```
Statement st = con.createStatement();
```

You can then execute the statement, get the ResultSet and iterate over it:

```
ResultSet rs = st.executeQuery("select * from employeeList");

while (rs.next()) {

  System.out.println(rs.getString("empname"));

}
```

PreparedStatement

PreparedStatement is a sub interface of the Statement interface. PreparedStatements are pre-compiled and hence their execution is much faster than that of Statements. You get a PreparedStatement object from a Connection object using the prepareStatement() method:

```
PreparedStatement    ps1    =    con.prepareStatement("insert    into
employeeList values ('Heartin',2)");

ps1.executeUpdate();
```

I can use any sql that I use in a Statement. One difference here is that in a Statement you pass the sql in the execute method, but in PreparedStatement you have to pass the sql in the prepareStatement() method while creating the PreparedStatement and leave the execute method empty. You can even override the sql statement passed in prepareStatement() by passing another one in the execute method, though it will not give the advantage of precompiling in a PreparedStatement.

`PreparedStatement also has a set of setXXX() methods, with which you can parameterize a PreparedStatement as:`

```
PreparedStatement    ps1    =    con.prepareStatement("insert    into
employeeList values (?,?)");
```

```
ps1.setString(1, "Heartin4");
```

```
ps1.setInt(2, 7);
```

```
ps1.executeUpdate();
```

This is very useful for inserting SQL 99 data types like BLOB and CLOB. Note that the index starts from 1 and not 0 and the setXXX() methods should be called before calling the executeUpdate() method.

CallableStatement

CallableStatement extends the capabilities of a PreparedStatement to include methods that are only appropriate for stored procedure calls; and hence CallableStatement is used to execute SQL stored procedures. Whereas PreparedStatement gives methods for dealing with IN parameters, CallableStatement provides methods to deal with OUT parameters as well.

Consider a stored procedure with signature as ‘updateID(oldid in int,newid in int,username out varchar2)’. *Code to create this stored proc is given in the end.* We can use a CallableStatement for executing it as:

```
CallableStatement cs=con.prepareCall("{call updateID(?,?,?)}");
```

```
cs.setInt(1, 2);
```

```
cs.setInt(2, 4);
```

```
cs.registerOutParameter(3, java.sql.Types.VARCHAR);
```

```
cs.executeQuery();
```

```
System.out.println(cs.getString(3));
```

If there are no OUT parameters, we can even use PreparedStatement. But using a CallableStatement is the right way to go for stored procedures. A CallableStatement can return one ResultSet object or multiple ResultSet objects. Multiple ResultSet objects are handled using operations inherited from Statement. You can use getResultSet() to get a result as a ResultSet. The getResultSet() method should be called only once per result. So we should use getMoreResults() to move to this Statement's next result, which returns true if it is a ResultSet object. If true, then we can call getResultSet() again to get the next result as a ResultSet object.

For maximum portability, a call's ResultSet objects and update counts should be processed prior to getting the values of output parameters. *Statement vs PreparedStatement vs CallableStatement is a favorite JDBC topic for many interviewers.*

ResultSet

This interface represents a table of data representing a database result set, which is usually generated by executing a statement that queries the database.

ParameterMetaData

ParameterMetaData retrieves the type and properties of the parameters used in the PreparedStatement interface. For some queries and driver implementations, the data that would be returned by a ParameterMetaData object may not be available until the PreparedStatement has been executed. Some driver implementations may even not be able to provide information about the types and properties for each parameter marker in a CallableStatement object. The below example however worked fine for me with Oracle thin driver and Oracle XE database and printed 3.

```
CallableStatement cs=con.prepareCall("{call updateID(?,?,?)}");


cs.setInt(1, 2);


cs.setInt(2, 4);


cs.registerOutParameter(3, java.sql.Types.VARCHAR);
```

```
ParameterMetaData paramMetaData = cs.getParameterMetaData();

System.out.println("Count="+paramMetaData.getParameterCount());

cs.executeQuery();
```

We can use ParameterMetaData with CallableStatement as well as in this example because CallableStatement is also a PreparedStatement.

ResultSetMetaData

ResultSetMetaData is used to retrieve information about the count, types and the properties of columns used in a ResultSet object. Consider an example where you are passed a ResultSet to a method and you don’t know the number or type of columns in the ResultSet. You can use ResultSetMetaData to get the column details and then find the corresponding row values using the column details.

```
public static void printColumnNames(ResultSet resultSet) throws
SQLException {

      if (resultSet != null) {

      ResultSetMetaData rsMetaData = resultSet.getMetaData();

      int numberOfColumns = rsMetaData.getColumnCount();

      for (int i = 1; i < numberOfColumns + 1; i++) {

      String columnName = rsMetaData.getColumnName(i);

      System.out.println("column name=" + columnName);
      }

  }

}
```

RowId

RowId maps a java object with a RowId. The RowId is a built-in datatype and is used as the identification key of a row in a database table, especially when there are duplicate rows.

```
 while (rs.next()) {

      System.out.println(rs.getString("columnName"));

      oracle.sql.ROWID rowid = (ROWID)rs.getRowId("columnName");

      System.out.println(rowid);

  }
```

We need to provide columnName or column index to getRowId same as rs.getString().

SQLException

This class is an exception class that provides information on a database access error or other errors. JDBC 4.0 introduced following refined subclasses of SQLException:

- java.sql.SQLClientInfoException
- java.sql.SQLDataException
- java.sql.SQLFeatureNotSupportedException
- java.sql.SQLIntegrityConstraintViolationException
- java.sql.SQLInvalidAuthorizationSpecException
- java.sql.SQLSyntaxErrorException

- java.sql.SQLTransactionRollbackException
- java.sql.SQLTransientConnectionException

There are still more interfaces which are part of JDBC core API (java.sql package). You can look through the documentation and try them all as part of learning or as required.

Database Code: updateID stored procedure used in examples

create or replace procedure updateID(oldid in int,newid in int,username out varchar2)

is

begin

update student set id=newid where id=oldid;

select firstname into username from student where id=newid;

commit;

end updateID;

How are tables related/connected to each other in a relational database?

How to get a connection in JDBC?

After you've installed the appropriate driver, it is time to establish a database connection using JDBC.

The programming involved to establish a JDBC connection is fairly simple. Here are these simple four steps −

Import JDBC Packages: Add import statements to your Java program to import required classes in your Java code.

Register JDBC Driver: This step causes the JVM to load the desired driver implementation into memory so it can fulfill your JDBC requests.

Database URL Formulation: This is to create a properly formatted address that points to the database to which you wish to connect.

Create Connection Object: Finally, code a call to the DriverManager object's getConnection( ) method to establish actual database connection.

Difference between Statement, PreparedStatement, and CallableStatement?

Once a connection is obtained we can interact with the database. The JDBC *Statement, CallableStatement,* and *PreparedStatement* interfaces define the methods and properties that enable you to send SQL or PL/SQL commands and receive data from your database. They also define methods that help bridge data type differences between Java and SQL data types used in a database.

The following table provides a summary of each interface's purpose to decide on the interface to use.

Interfaces	**Recommended Use**
Statement	Use the for general-purpose access to your database. Useful when you are using static SQL statements at runtime. The Statement interface cannot accept parameters.
PreparedStatement	Use the when you plan to use the SQL statements many times. The PreparedStatement interface accepts input parameters at runtime.

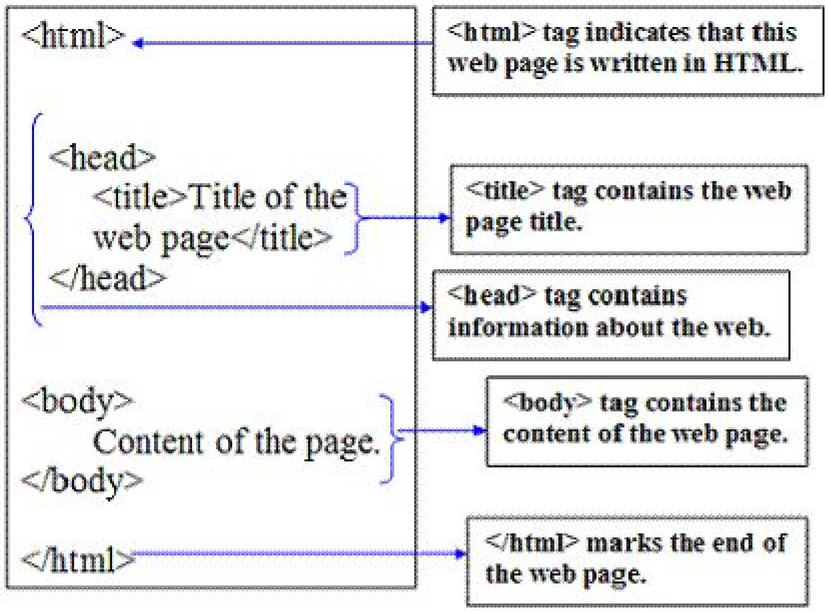
CallableStatement	Use the when you want to access the database stored procedures. The CallableStatement interface can also accept runtime input parameters.

HTML/CSS

What is HTML

Hypertext Markup Language is the standard markup language for creating web pages and web applications. With Cascading Style Sheets and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.

What is the structure of an HTML document? List some tags. What is <head> used for? <body>?



<html>
<head>
<title>Title of the web page</title>
</head>
<body>
Content of the page.
</body>
</html>
<html> tag indicates that this web page is written in HTML.
<title> tag contains the web page title.
<head> tag contains information about the web.
<body> tag contains the content of the web page.
</html> marks the end of the web page.

What is a doctype?

The <!DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in.

What is the tag for an ordered list? Unordered list?

<ul> & <ol>

What are some HTML5 tags? Why were HTML5 tags introduced?

Tag	Description
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	Defines information about the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
<br>	Inserts a single line break
<hr>	Defines a thematic change in the content
<!--...-->	Defines a comment

Do all tags come in a pair? What are the other things inside tags called? list some.

Not all tags come in pairs, such as <br> and <img>.   The elements that can be included within a tag include attributes such as “Style”,”Alt” & “Href”.

What is the syntax for a comment in HTML?

<!-- -->

Give me the HTML markup for a table.

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

What are some tags you would use in a form?

```
<form>
  <input type="radio" name="gender" value="male" checked>
Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

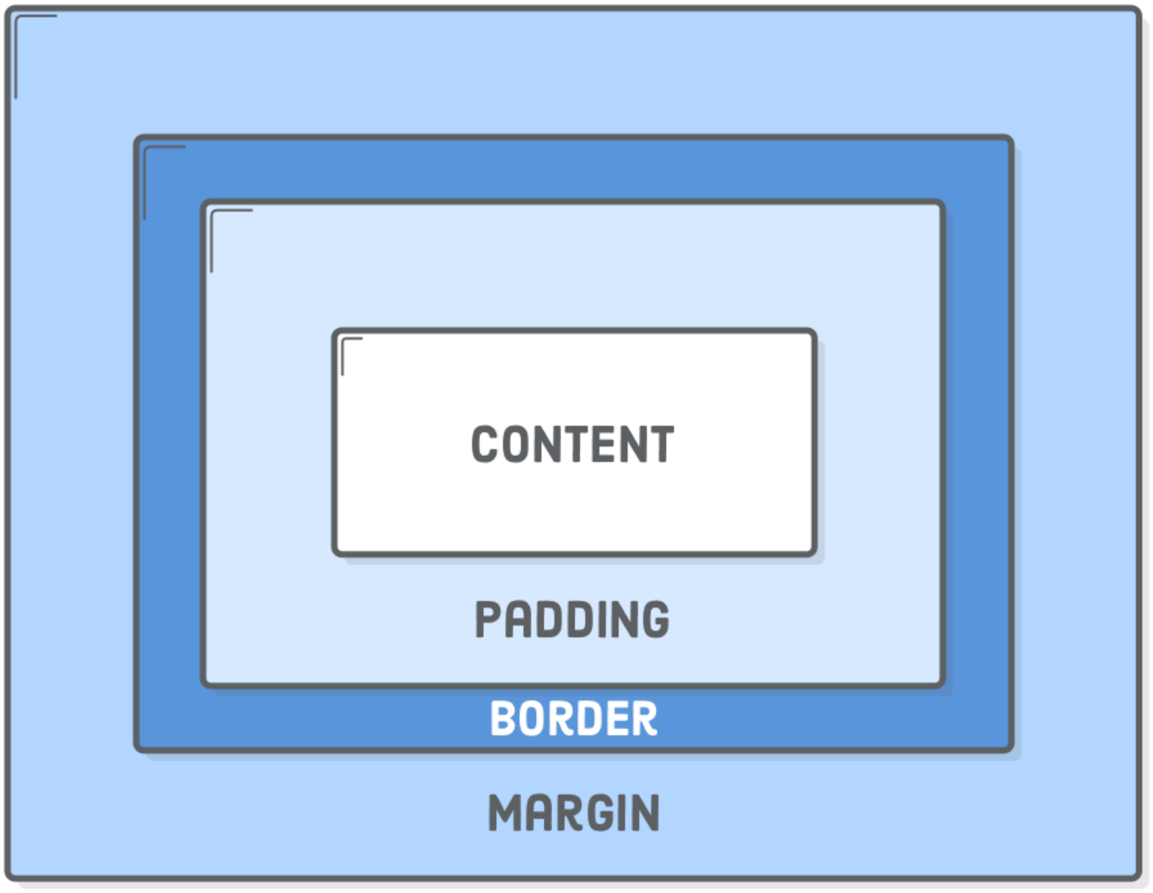
What is CSS? what are the different ways of styling an HTML file? Which is best? Why?

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

The industry standard is to style an HTML page using links to an external style sheet. This makes it possible to style multiple pages using class and ID tags.

Describe the CSS box model.

In web development, the CSS box model refers to the composition of a webpage or HTML page from blocks or boxes using CSS. Specifically, the box model describes how the size of each such block and its content is determined through styling instructions.



Which way has highest priority when styles cascade: inline, internal, and external.

Inline takes top priority, whereas internal takes priority over external.

Syntax for styling an element? What is a class and how to style them? What is an id? how to style? Difference?

To style an element, you can either use inline styling with <div style=”color:red; width:100px”> or you can define an ID or Class and style that externally, for example <div class=”MyStyle>

The difference between a DIV and an ID is that a ID is for an individual element, whereas a class is for groups of elements. To address a class in your CSS document, you’d add a . for example .MyStyle {}, for an ID you’d add a hashtag, for example #MyStyle {}

What if I want to select child elements, What syntax is that?

In CSS, selectors are patterns used to select the element(s) you want to style.

Selector	**Example**	**Example description**
.class	.intro	Selects all elements with class="intro"
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element	div, p	Selects all <div> elements and all <p> elements
element element	div p	Selects all <p> elements inside <div> elements
element>element	div > p	Selects all <p> elements where the parent is a <div> element

element+element	div + p	Selects all <p> elements that are placed immediately after <div> elements
element1~element2	p ~ ul	Selects every <ul> element that are preceded by a <p> element
[*attribute*]	[target]	Selects all elements with a target attribute
[*attribute=value*]	[target=_blank]	Selects all elements with target="_blank"
[*attribute~=value*]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[*attribute|=value*]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"
[*attribute^=value*]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[*attribute$=value*]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[*attribute*=value*]	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"
:active	a:active	Selects the active link
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element

:checked	input:checked	Selects every checked <input> element
:default	input:default	Selects the default <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children (including text nodes)
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> element that is the first child of its parent
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the input element which has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects input elements with a value within a specified range
:indeterminate	input:indeterminate	Selects input elements that are in an indeterminate state

:invalid	input:invalid	Selects all input elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)
:last-child	p:last-child	Selects every <p> element that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(*selector*)	:not(p)	Selects every element that is not a <p> element
:nth-child(*n*)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(*n*)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(*n*)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent

:optional	input:optional	Selects input elements with no "required" attribute
:out-of-range	input:out-of-range	Selects input elements with a value outside a specified range
::placeholder	input::placeholder	Selects input elements with placeholder text
:read-only	input:read-only	Selects input elements with the "readonly" attribute specified
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified
:required	input:required	Selects input elements with the "required" attribute specified
:root	:root	Selects the document's root element
::selection	::selection	Selects the portion of an element that is selected by a user
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all input elements with a valid value
:visited	a:visited	Selects all visited links

Can I select multiple elements at once? How?

To select multiple elements at once, you can use the .element selector, or assign a class to everything you'd like to change, and then define the class in your external style sheet.

What is Bootstrap?

Bootstrap is a free and open-source front-end framework for developing websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

Describe the Bootstrap grid system

Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases. It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.

JavaScript

What is JavaScript? What do we use it for?

JavaScript, often abbreviated as JS, is a high-level, interpreted programming language that conforms to the ECMAScript specification. It is a language that is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Where is the best place to put a script tag in your HTML document?

The head of the document

What are the data types in JS?

Number, String, Boolean, Null, Undefined, Object, Symbol

What are the variable scopes in JS?

Scope in JavaScript refers to the current context of code, which determines the accessibility of variables to JavaScript. The two types of scope are local and global: Global variables are those declared outside of a block.

What are JS objects? What is the syntax?

JavaScript is designed on a simple object-based paradigm. An object is a collection of properties, and a property is an association between a name (or key) and a value. A property's value can be a function, in which case the property is known as a method.

```
var myCar = new Object();
myCar.make = 'Ford';
myCar.model = 'Mustang';
myCar.year = 1969;
```

What is JSON? is it different from JS objects?

JSON stands for JavaScript Object Notation

JSON is a lightweight format for storing and transporting data

JSON is often used when data is sent from a server to a web page

JSON is "self-describing" and easy to understand

JSON Objects

JSON objects are written inside curly braces.

Just like in JavaScript, objects can contain multiple name/value pairs:

{"firstName":"John", "lastName":"Doe"}

The JSON values can only be one of the six data types (strings, numbers, objects, arrays, Boolean, null). JavaScript values on the other hand can be any valid JavaScript Structure. Unlike JavaScript Object, a JSON Object has to be fed into a variable as a String and then parsed into JavaScript.

How to convert JS object to/from JSON?

A common use of JSON is to exchange data to/from a web server.

When sending data to a web server, the data has to be a string.

Convert a JavaScript object into a string with JSON.stringify().

What does truthy/falsy mean?

In JavaScript, a truthy value is a value that is considered true when encountered in a Boolean context. All values are truthy unless they are defined as falsy (i.e., except for false , 0 , "" , null , undefined , and NaN). JavaScript uses type coercion in Boolean contexts.

What is type coercion?

Since JavaScript is a weakly-typed language, values can also be converted between different types automatically, and it is called implicit type coercion. It usually happens when you apply operators to values of different types

What is the difference between == and === ?

JavaScript has both strict and type-converting equality comparison. For strict equality the objects being compared must have the same type and:

- Two strings are strictly equal when they have the same sequence of characters, same length, and same characters in corresponding positions.

- Two numbers are strictly equal when they are numerically equal (have the same number value). NaN is not equal to anything, including NaN. Positive and negative zeros are equal to one another.

- Two Boolean operands are strictly equal if both are true or both are false.

- Two objects are strictly equal if they refer to the same Object.

- Null and Undefined types are == (but not ===). [I.e. (Null==Undefined) is true but (Null===Undefined) is false]

How does inheritance work in JS?

JavaScript does not have "methods" in the form that class-based languages define them. In JavaScript, any function can be added to an object in the form of a property. An inherited function acts just as any other property

What is unique about functions in JS?

Image result for functions in javascriptcodeburst.io
Like the program itself, a function is composed of a sequence of statements called the function body. Values can be passed to a function, and the function will return a value. In JavaScript, functions are first-class objects, because they can have properties and methods just like any other object.

What are callback functions? self-invoking functions?

<u>Callback Functions</u>

In JavaScript, functions are objects. Because of this, functions can take functions as arguments, and can be returned by other functions. Functions that do this are called higher-order functions. Any function that is passed as an argument is called a callback function.

<u>Self-Invoking Functions</u>

Function expressions can be made "self-invoking".

A self-invoking expression is invoked (started) automatically, without being called.

Function expressions will execute automatically if the expression is followed by ().

You cannot self-invoke a function declaration.

You have to add parentheses around the function to indicate that it is a function expression

What is closure? Why use closures?

A closure is a feature in JavaScript where an inner function has access to the outer (enclosing) function's variables — a scope chain.

The closure has three scope chains:

- it has access to its own scope — variables defined between its curly brackets
- it has access to the outer function's variables
- it has access to the global variables

At first glance, a closure is simply a function defined within another function. However, the power of closures is derived from the fact that the inner function remembers the environment in which it was created. In other words, the inner function has access to the outer function's variables and parameter.

One powerful use of closures is to use the outer function as a factory for creating functions that are somehow related.

What is the this keyword?

The JavaScript `this` keyword refers to the object it belongs to.

It has different values depending on where it is used:

In a method, `this` refers to the owner object.
Alone, `this` refers to the global object.
In a function, `this` refers to the global object.
In a function, in strict mode, `this` is `undefined`.
In an event, `this` refers to the element that received the event.

Methods like `call()`, and `apply()` can refer `this` to any object.

Does JS have classes? when were they introduced?

ES6 introduced the class syntax, but it is only syntactic sugar for prototype inheritance

What is new with ES6? What are template literals, let keyword, Symbols, Promises?

Template literals are **string literals** allowing embedded expressions. You can use multi-line strings and **string** interpolation features with them. They were called "**template** strings" in prior editions of the ES2015 specification.

let allows you to declare variables that are limited in scope to the block, statement, or **expression** on which it is used. This is unlike the var keyword, which defines a **variable** globally, or locally to an entire function regardless of block scope

A key part of what makes Symbols useful, is a set of Symbol constants, known as “well known symbols”. These are effectively a bunch of static properties on the Symbol **class** which are implemented within other native objects, such as Arrays, Strings, and within the internals of the JavaScript engine.

Promises provide a simpler alternative for executing, composing, and managing asynchronous operations when compared to traditional callback-based approaches. They also allow you to handle asynchronous errors using approaches that are similar to synchronous try/catch .

Arrow function? What makes it different from normal function?

Arrow functions – also called “fat arrow” functions, from CoffeeScript (a transcompiled language) — are a more concise syntax for writing function expressions. ... They are one-line mini functions which work much like Lambdas in other languages like C# or Python.

What is strict mode?

Strict mode in JavaScript. **Strict Mode** is a new feature in ECMAScript 5 that allows you to place a program, or a function, in a “**strict**” operating context. This **strict** context prevents certain actions from being taken and throws more exceptions.

What will happen when I run this code: console.log(0.1+0.2==0.3) ?

The console will return false.

What are arrays in JS? can you change their size?

The **Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An **array** is used to store a collection of data, but it is often more useful to think of an **array** as a collection of variables of the same type.

What is a CDN? What are the benefits?

A content delivery network or content distribution network is a geographically distributed network of proxy servers and their data centers. The goal is to provide high availability and high performance by distributing the service spatially relative to end-users.

What is Fetch? Why do we use it?

The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses. It also provides a global fetch() method that provides an easy, logical way to fetch resources asynchronously across the network.

What are the steps for sending a Fetch request?

First of all, we invoke the fetch() method, passing it the URL of the resource we want to fetch. This is the modern equivalent of request.open() in XHR, plus you don't need any equivalent to .send().

After that, you can see the .then() method chained onto the end of fetch() — this method is a part of Promises, a modern JavaScript feature for performing asynchronous operations. fetch() returns a promise, which resolves to the response sent back from the server — we use .then() to run some follow-up code after the promise resolves, which is the function we've defined inside it. This is the equivalent of the onload event handler in the XHR version.

This function is automatically given the response from the server as a parameter when the fetch() promise resolves. Inside the function we grab the response and run its text()method, which basically returns the response as raw text. This is the equivalent of request.responseType = 'text' in the XHR version.

You'll see that text() also returns a promise, so we chain another .then() onto it, inside of which we define a function to receive the raw text that the text() promise resolves to.

Inside the inner promise's function, we do much the same as we did in the XHR version — set the <pre> element's text content to the text value.

What is AJAX? why do we use it?

Ajax is a set of Web development techniques using many web technologies on the client side to create asynchronous Web applications. With Ajax, web applications can send and retrieve data from a server asynchronously without interfering with the display and behavior of the existing page.

What are steps to sending an AJAX request?

```
var GetCharData = function() { var queue = [], running = false, next =
function() { running = false; if (queue.length) GetCharData(queue.shift()); };
return function(me) { if (running) return queue.push(me); running = true;
$.ajax({ url: 'allmoves.php?path=getchart&'+me.attr('id'), cache: false,
success: function(msg) { // Handle Success }, error: function(msg) { // Handle
Failure }, complete: next }); }; }();
```

Then:

```
$(".miniCanvas").each(function() { GetCharData($(this)); })
```

What is the DOM?

The Document Object Model is a cross-platform and language-independent application programming interface that treats an HTML, XHTML, or XML document as a tree structure wherein each node is an object representing a part of the document. The DOM model represents a document with a logical tree.

How to select elements from the DOM? How to insert elements dynamically?

You can access a <select> element by using getElementById():

You can create a <select> element by using the document.createElement() method

What are events / event listeners? What are some events we can listen for? ways of setting event listeners?

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

An event listener is a procedure or function in a computer program that waits for an event to occur.

The `addEventListener()` method attaches an event handler to the specified element.

The `addEventListener()` method attaches an event handler to an element without overwriting existing event handlers.

You can add many event handlers to one element.

You can add many event handlers of the same type to one element, i.e two "click" events.

You can add event listeners to any DOM object not only HTML elements. i.e the window object.

The `addEventListener()` method makes it easier to control how the event reacts to bubbling.

When using the `addEventListener()` method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.

You can easily remove an event listener by using the removeEventListener() method.

What is bubbling and capturing and what is the difference?

Event bubbling and capturing are two ways of event propagation in the HTML DOM API, when an event occurs in an element inside another element, and both elements have registered a handle for that event. The event propagation mode determines in which order the elements receive the event.

With bubbling, the event is first captured and handled by the innermost element and then propagated to outer elements.

With capturing, the event is first captured by the outermost element and propagated to the inner elements.

Capturing is also called "trickling", which helps remember the propagation order:

trickle down, bubble up

Example:

```
<div>

        <ul>

                <li></li>

        </ul>

 </div>
```

In the structure above, assume that a click event occurred in the li element.

In capturing model, the event will be handled by the div first (click event handlers in the div will fire first), then in the ul, then at the last in the target element, li.

In the bubbling model, the opposite will happen: the event will be first handled by the li, then by the ul, and at last by the div element.

How do you stop an event from bubbling further?

use event.stopPropogation()

Servlets (part one)

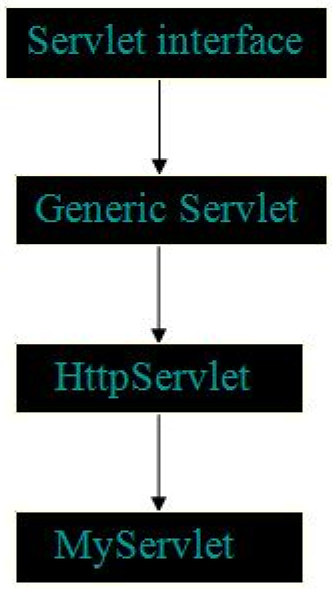
What is a servlet?

A Java servlet is a Java software component that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement web containers for hosting web applications on web servers and thus qualify as a server-side servlet web API.

What is the hierarchy of servlets?

The Servlet interface is the root interface of the servlet class hierarchy. All Servlets need to either directly or indirectly implement the Servlet interface. The GenericServlet class of the Servlet API implements the Servlet interface. In addition to the Servlet interface, the GenericServlet class implements the ServletConfig interface of the Servlet API and the Serializable interface of the standard java.io. package. The object of the ServletConfig interface is used by the Web container to pass the configuration information to a servlet when a servlet is initialized.

To develop a servlet that communicates using HTTP, we need to extend the HttpServlet class in our servlet. The HttpServlet class extends the GenericServlet class and provides built-in HTTP functionality.



What is the lifecycle of a servlet?

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

1. The servlet is initialized by calling the init() method.
2. The servlet calls service() method to process a client's request.
3. The servlet is terminated by calling the destroy() method.
4. Finally, servlet is garbage collected by the garbage collector of the JVM.

The init() Method

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.

The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.

When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet

The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

The destroy() Method

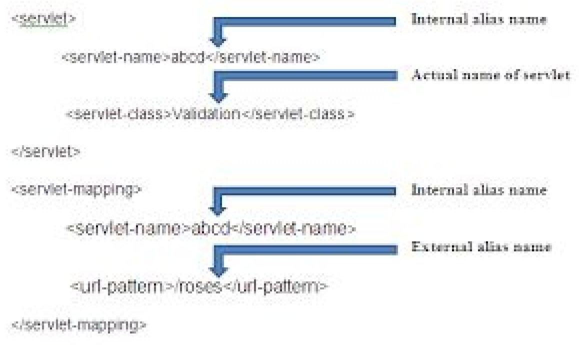
The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

After the destroy() method is called, the servlet object is marked for garbage collection.

Describe the deployment descriptor

A deployment descriptor (DD) refers to a configuration file for an artifact that is deployed to some container/engine. In the Java Platform, Enterprise Edition, a deployment descriptor describes how a component, module or application (such as a web application or enterprise application) should be deployed.

What are some key tags in the deployment descriptor?


```
<servlet>
    <servlet-name>abcd</servlet-name>
    <servlet-class>Validation</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>abcd</servlet-name>
    <url-pattern>/roses</url-pattern>
</servlet-mapping>
```

How do I map a Servlet?

Servlets should be registered with servlet container. For that, you should add entries in web deployment descriptor web.xml. It is located in WEB-INF directory of the web application. Entries to be done in web.xml for servlet-mapping:

```
<servlet-mapping>
<servlet-name>milk</servlet-name>
<url-pattern>/drink/*</url-pattern>
</servlet-mapping>
```

servlet-mapping has two child tags, url-pattern and servlet-name. url-pattern specifies the type of urls for which, the servlet given in servlet-name should be called. Be aware that, the container will use case-sensitive for string comparisons for servlet matching.

What are the doGet and doPost methods?

The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

The doPost() Method

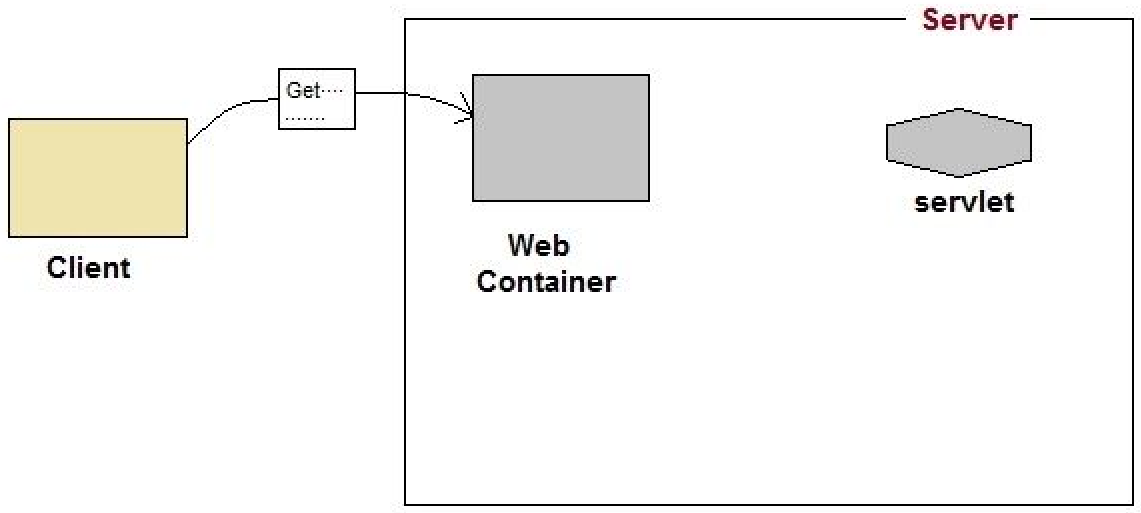
A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

What is the method signature of doGet and doPost?

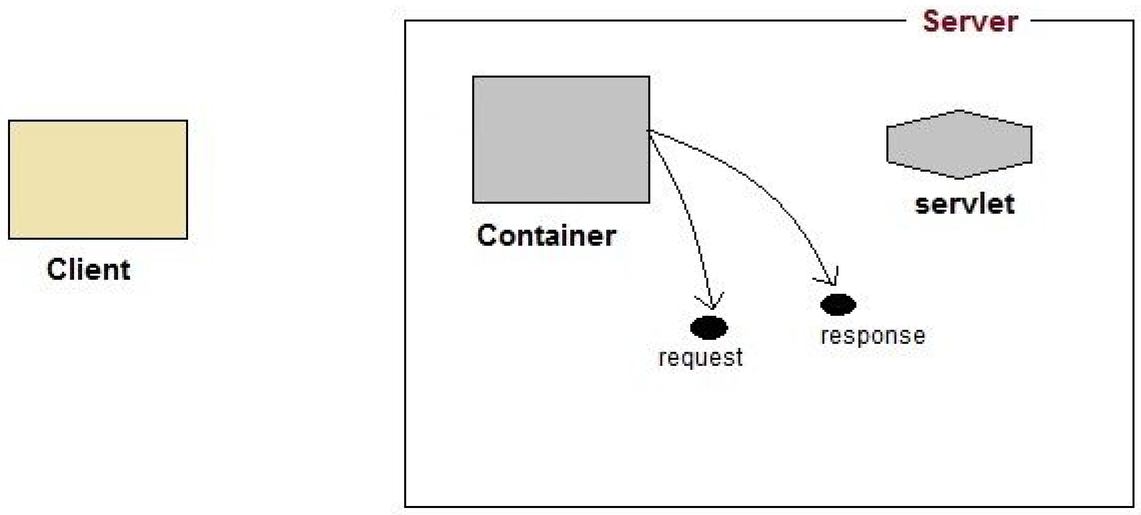
```
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}

public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}
```

Describe the flow of a request with servlets

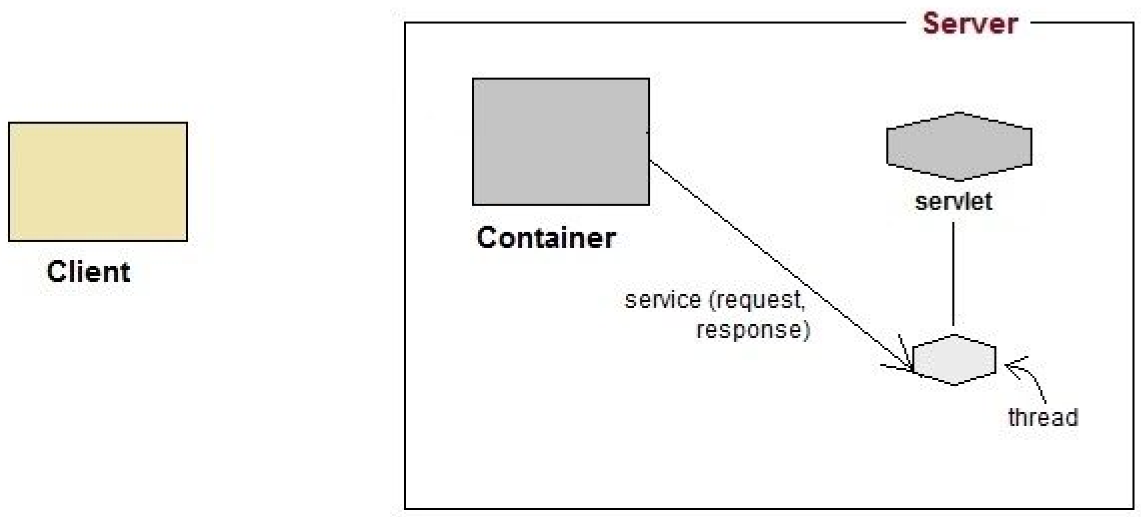
1. User sends request for a servlet by clicking a link that has URL to a servlet.



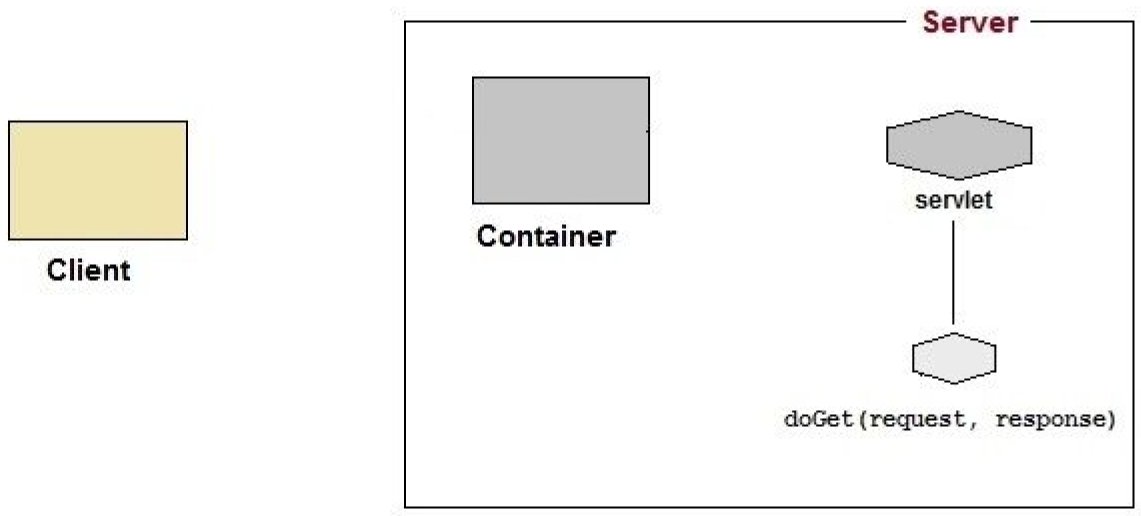
2. The container finds the servlet using deployment descriptor and creates two objects :
   A. HttpServletRequest
   B. HttpServletResponse



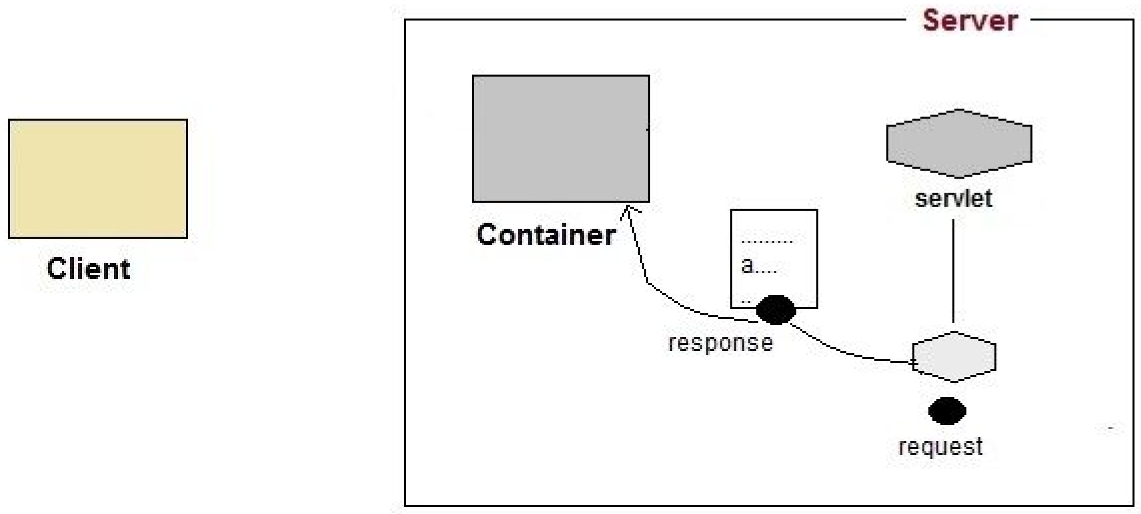
3. Then the container creates or allocates a thread for that request and calls the Servlet's `service()`method and passes the **request, response** objects as arguments.



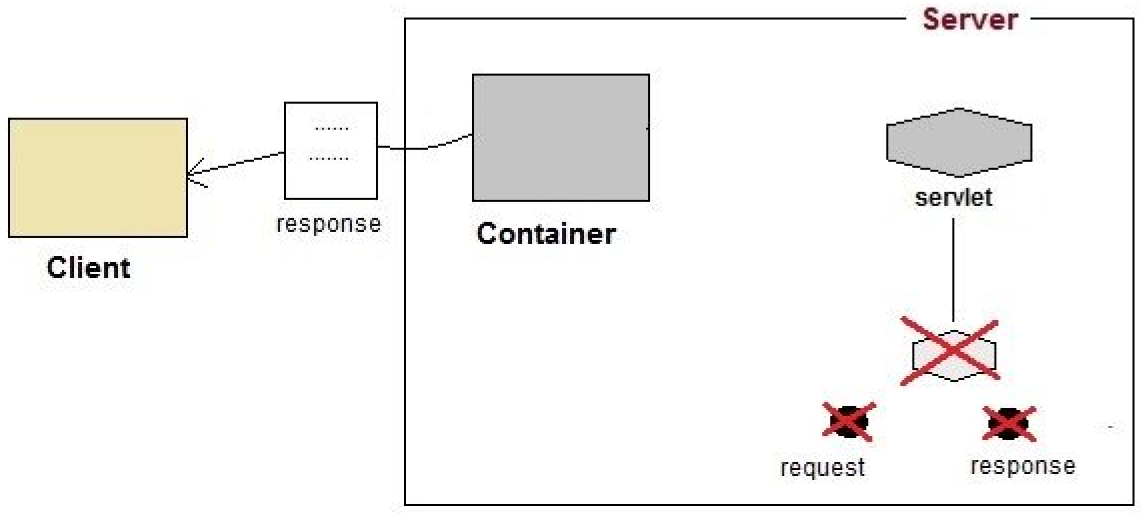
4. The `service()` method, then decides which servlet method, `doGet()` or `doPost()` to call, based on **HTTP Request Method**(Get, Post etc) sent by the client. Suppose the client sent an HTTP GET request, so the `service()` will call Servlet's `doGet()` method.



5. Then the Servlet uses response object to write the response back to the client.



6. After the `service()` method is completed the **thread** dies. And the request and response objects are ready for **garbage collection**.



Servlets (part two)

By default, when is a servlet first instantiated and loaded onto the Servlet Container?

When the servlet container (like Apache Tomcat) starts up, it will deploy and load all its web applications. When a web application is loaded, the servlet container creates the ServletContext once and keeps it in the server's memory. The web app's web.xml file is parsed, and each <servlet>, <filter> and <listener> found (or each class annotated with @WebServlet, @WebFilter and @WebListener respectively) is instantiated once and kept in the server's memory as well. For each instantiated filter, its init() method is invoked with a new FilterConfig.

When the servlet container shuts down, it unloads all web applications, invokes the destroy() method of all its initialized servlets and filters, and all ServletContext, Servlet, Filter and Listener instances are trashed.

How can a servlet be instantiated when the web app starts up?

When the servlet container (like Apache Tomcat) starts up, it will deploy and load all its web applications. When a web application is loaded, the servlet container creates the `ServletContext`once and keeps it in the server's memory. The web app's `web.xml` file is parsed, and each `<servlet>`, `<filter>` and `<listener>` found (or each class annotated with `@WebServlet`, `@WebFilter` and `@WebListener` respectively) is instantiated once and kept in the server's memory as well. For each instantiated filter, its `init()` method is invoked with a new `FilterConfig`.
When the servlet container shuts down, it unloads all web applications, invokes the `destroy()`method of all its initialized servlets and filters, and all `ServletContext`, `Servlet`, `Filter` and `Listener` instances are trashed.

When a `Servlet` has a `<servlet><load-on-startup>` or `@WebServlet(loadOnStartup)` value greater than `0`, its `init()` method is also invoked during startup with a new `ServletConfig`. Those servlets are initialized in the same order specified by that value (1 -> 1st, 2 -> 2nd, etc). If the same value is specified for more than one servlet, then each of those servlets is loaded in the order they appear in the `web.xml`, or `@WebServlet` classloading. In the event the "load-on-startup" value is absent, the `init()` method will be invoked whenever the HTTP request hits that servlet for the very first time.

What is a redirect?

The sendRedirect() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, jsp or html file. It accepts relative as well as absolute URL. It works at client side because it uses the url bar of the browser to make another request.

What is a forward?

public interface **RequestDispatcher**

Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server. The servlet container creates the

RequestDispatcher object, which is used as a wrapper around a server resource located at a particular path or given by a particular name.

This interface is intended to wrap servlets, but a servlet container can create RequestDispatcher objects to wrap any type of resource.

How to perform a redirect with Servlets? How is it different from performing a forward? (DETAILS, please)

<u>Send redirect servlet</u>
The sendRedirect() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, jsp or html file. It accepts relative as well as absolute URL. It works at client side because it uses the url bar of the browser to make another request.

What is a cookie?

An HTTP cookie is a small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing. Cookies were designed to be a reliable mechanism for websites to remember stateful information or to record the user's browsing activity.

Describe the use of HttpSession for user session management. What other tools are there?

Web Session comprises of the sequence of HTTP requests and responses bounded to a user using a resource over a fixed timestamp. It can be described as storage of information by the server that persists over user interactions with the web application.

HTTP was designed to transfer documents and is not an application transport protocol. It is a stateless protocol as it does not store any record of its previous interaction. Each interaction is independent of any other web interactions and is based on the information contained in the HTTP Headers. In persistent HTTP connection socket is left open using Alive parameters as "Connection: keep-alive" for the longer duration but still have timeout value so as not to overload server. Multiple requests are pipeline where each request is independent of each other. The higher the timeout, the more server processes will be kept occupied waiting on connections with idle clients. Even if the socket is kept open, HTTP is stateless as it doesn't store anything. Moreover, both documents and applications are text based but the application requires maintenance of state through some way.

Present day advanced web applications demand preserving the status of users over the duration of multiple requests. They desire a technique to identify ever user connection. HTTP is stateless, so it seems obvious that HTTP would not be appropriate for delivering applications. Additionally, sessions are coupled with connections, but they work at different OSI layer and have different timeout values. For the Apache the default value for which session remains in memory is 300 seconds, also connection will be closed once its idle for 60 seconds. Connection timeout can't be increased as it will possible exhaust TCP connection limit, will affect memory as well. The difficulty lies in binding Session and Connection. Moreover, Sessions will remain in the memory of server even after its associated connection has been terminated due to inactivity. Implanting cookies on clients system will enable the application to find the session on the server. Through the exchange of session IDs, the state can be maintained even for a stateless protocol like HTTP.

With the use of sessions and cookies, HTTP was given the means by which state could be tracked throughout the use of an application.

Thereafter session management is required to establish a link connecting authentication and access control. While the server stores session information, Session ID is used to identify a session and is mostly stored in the cookie on client system formed at the time of session creation. Once the user has been authenticated the application can make use of sessions. Web application creates the session in order to keep track of users, identify users on successive request, authorizing user on the different domain, applying access control mechanism, accounting and tracking user’s activity moreover to increase the usability of the application. After successful authentication and establishment of the web session, Session ID or token is used as a method for authenticating further requests for the current session. Session Id or token binds the user HTTP traffic to its permissible access rights. HTTP uses various mechanisms to maintain session state for web applications such as cookies, URL rewriting, URL arguments on Getting requests, body arguments on POST requests.

Session ID length must be long enough, combining random number so as to avoid its attacks like replay, brute force. Even if it is decoded it should not reveal any personally identifiable information like password, secret information etc. Moreover its name should not reveal information about its purpose it serves. The disclosure, capture, guessing, prediction, brute force, or fixation of the session ID will lead to session hijacking attacks, where an attacker can completely imitate a victim user in the web application. The intruder can conduct either targeted attack on a specific user or generic attack impersonating any legitimate user.

Web development frameworks, such as J2EE, ASP.NET, PHP, and others provide their own session management features. However they contain inherent vulnerabilities and weaknesses, so it is always recommended to use the latest available version available, that has possibly fixed all the well-known vulnerabilities.

Conclusion

Session Management is a link connecting authentication and access control. Many modern day networking devices provides an ability to control, directly manipulate and manage any IP application traffic. Furthermore encrypting, digitally signing and implanting browser cookies by manipulating HTTP traffic.

What is the difference between ServletConfig and ServletContext?

ServletConfig is used for sharing init parameters specific to a servlet while ServletContext is for sharing init parameters within any Servlet within a web application. ServletContext interface represents Servlets view of the Web Application it belongs to. ServletContext present within ServletConfig

How are parameters for ServletConfig and ServletContext (a) declared in application setup, and (b) accessed in Java code?

ServletConfig

- `ServletConfig` available in `javax.servlet.*;` package
- `ServletConfig` object is one per servlet class
- Object of `ServletConfig` will be created during initialization process of the servlet
- This Config object is public to a particular servlet only
- Scope: As long as a servlet is executing, `ServletConfig` object will be available, it will be destroyed once the servlet execution is completed.
- We should give request explicitly, in order to create `ServletConfig` object for the first time
- In web.xml – `<init-param>` tag will be appear under `<servlet-class>` tag

Here's how it looks under web.xml : (Source)

```
<servlet> <servlet-name>ServletConfigTest</servlet-name>
<servlet-class>com.stackoverflow.ServletConfigTest</servlet-class> <init-param>
<param-name>topic</param-name> <param-value>Difference between ServletConfig
and ServletContext</param-value> </init-param> </servlet>
```

ServletContext

- `ServletContext` available in `javax.servlet.*;` package
- `ServletContext` object is global to entire web application
- Object of `ServletContext` will be created at the time of web application deployment
- *Scope*: As long as web application is executing, `ServletContext` object will be available, and it will be destroyed once the application is removed from the server.
- `ServletContext` object will be available even before giving the first request In `web.xml` – `<context-param>` tag will be appear under `<web-app>` tag

Here's how it looks under web.xml :

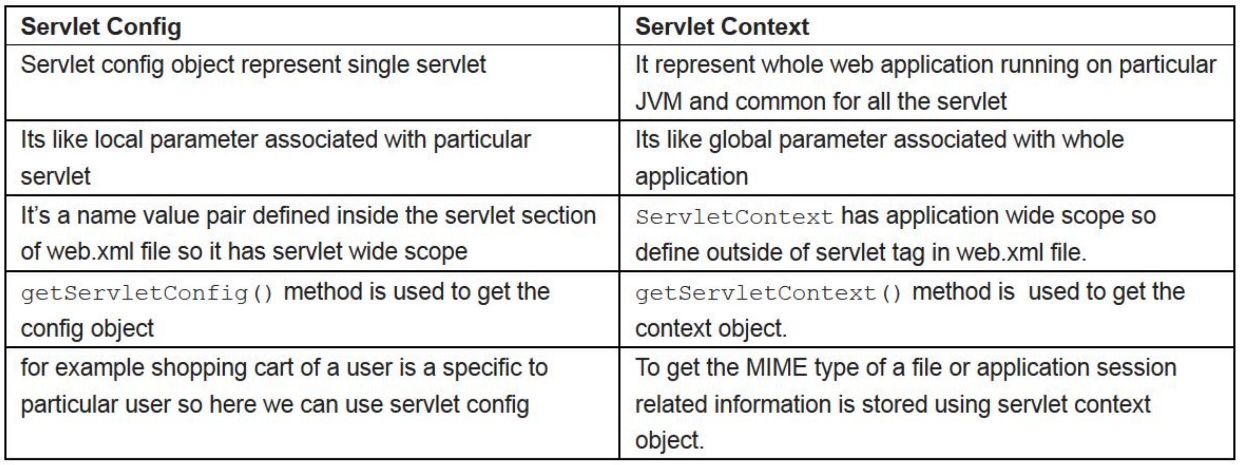
```
<context-param> <param-name>globalVariable</param-name>
<param-value>com.stackoverflow</param-value> </context-param>
```

So finally…….

No. of web applications = That many number of `ServletContext` objects [1 per web application]

No. of servlet classes = That many number of `ServletConfig` objects

Difference between ServletContext and ServletConfig in Servlets JSP in tabular format



Servlet Config	Servlet Context
Servlet config object represent single servlet	It represent whole web application running on particular JVM and common for all the servlet
Its like local parameter associated with particular servlet	Its like global parameter associated with whole application
It's a name value pair defined inside the servlet section of web.xml file so it has servlet wide scope	`ServletContext` has application wide scope so define outside of servlet tag in web.xml file.
`getServletConfig()` method is used to get the config object	`getServletContext()` method is used to get the context object.
for example shopping cart of a user is a specific to particular user so here we can use servlet config	To get the MIME type of a file or application session related information is stored using servlet context object.

Describe how to create a servlet (a) with annotations and (b) with XML configuration.

Annotations vs. Deployment Descriptor

What's the difference? Well, obviously the deployment descriptor is a separate file where you set configuration values in XML format, where the annotation is directly embedded in your source code. Use annotations if you prefer to have code and configuration at the same place for better readability. Deployment descriptors are the exact opposite – you separate code and configuration. This way you do not need to recompile the entire project if you want to change a single configuration value.

Angular

What is Node.js?

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser.

What is NPM?

npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry.

What is the package.json?

All npm packages contain a file, usually in the project root, called package.json - this file holds various metadata relevant to the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies.

What is bundling and what tool do we use to do it?

CLI npm build

Describe transpiling.

TypeScript is a primary language for Angular application development. It is a superset of JavaScript with design-time support for type safety and tooling. Browsers can't execute TypeScript directly. Typescript must be "transpiled" into JavaScript using the tsc compiler, which requires some configuration.

What is Angular?

AngularJS is a JavaScript-based open-source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications.

What is dependency injection?

In software engineering, dependency injection is a technique whereby one object supplies the dependencies of another object. A dependency is an object that can be used. An injection is the passing of a dependency to a dependent object that would use it. The service is made part of the client's state.

Where is the app folder located? What files does it contain if the project was created by the Angular CLI?

src/app, it’s located wherever the CLI was run. Within the folder, the structure looks like this

- app.module.ts
- app.component.ts
- component1
    component1.component.ts
- component2
    Component2.component.ts

In actuality each of those files is then broken down into their .html and .scss counterparts.

What is a decorator?

Decorators are a design pattern that is used to separate modification or decoration of a class without modifying the original source code. In AngularJS, decorators are functions that allow a service, directive or filter to be modified prior to its usage.

What is a module?

In Angular, a module is a mechanism to group components, directives, pipes and services that are related, in such a way that can be combined with other modules to create an application. An Angular application can be thought of as a puzzle where each piece (or each module) is needed to be able to see the full picture.

What is a component?

In AngularJS, a Component is a special kind of directive that uses a simpler configuration which is suitable for a component-based application structure. This makes it easier to write an app in a way that's similar to using Web Components or using the new Angular's style of application architecture.

What is @Component? What are some attributes you can find inside the decorator?

The @Component decorator identifies the class immediately below it as a component class, and specifies its metadata.

How would you add a new component without the Angular CLI?

Create the component.ts file inside the src folder.
Add the "@component" decorator
Import the component into the app.module.ts file.
Add a reference to the imported component into the "@NgModule" declarations.

What is a directive? What are the different types?

Angular directives are used to extend the power of the HTML by giving it new syntax. Each directive has a name — either one from the Angular predefined like ng-repeat , or a custom one which can be called anything. And each directive determines where it can be used: in an element , attribute , class or comment

There are three kinds of directives in Angular:

Components—directives with a template.
Structural directives—change the DOM layout by adding and removing DOM elements.
Attribute directives—change the appearance or behavior of an element, component, or another directive.

What is data binding? What are some different types?

Data binding is a core concept in Angular and allows to define communication between a component and the DOM, making it very easy to define interactive applications without worrying about pushing and pulling data. There are four forms of data binding and they differ in the way the data is flowing.

There are different types of data binding:

From the Component to the DOM

From the DOM to the Component

Two-way

What is a pipe in Angular? How do you create a custom pipe?

It is a way to write display-value transformations that you can declare in your HTML. You create a custom pipe by using the @pipe() on a class, implementing the PipeTransform. To you use it, you must also add it to a module.

What is a SPA?

A single-page application (SPA) is a web application or web site that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server. Interaction with the single page application often involves dynamic communication with the web server behind the scenes.

What is a service in Angular? (DETAILS, please)

AngularJS services are substitutable objects that are wired together using dependency injection (DI). You can use services to organize and share code across your app. AngularJS services are: Lazily instantiated – AngularJS only instantiates a service when an application component depends on it.

What is routing? What is it used for? (DETAILED steps, please)

If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.

The ngRoute module routes your application to different pages without reloading the entire application

To make your applications ready for routing, you must include the AngularJS Route module:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-route.js"></script>
```
Then you must add the ngRoute as a dependency in the application module:

```
var app = angular.module("myApp", ["ngRoute"]);
```
Now your application has access to the route module, which provides the $routeProvider.

Use the $routeProvider to configure different routes in your application:

```
app.config(function($routeProvider) {
  $routeProvider
  .when("/", {
    templateUrl : "main.htm"
  })
  .when("/red", {
    templateUrl : "red.htm"
  })
  .when("/green", {
    templateUrl : "green.htm"
  })
  .when("/blue", {
    templateUrl : "blue.htm"
  });
});
```

Your application needs a container to put the content provided by the routing.

This container is the ng-view directive.

You can use: <ng-view></ng-view>   OR   <div ng-view></div> OR <div class="ng-view"></div>

Applications can only have one ng-view directive, and this will be the placeholder for all views provided by the route.

How do you perform HTTP requests in Angular? (DETAILED steps, please)

The HttpClient in @angular/common/http offers a simplified client HTTP API for Angular applications that rests on the XMLHttpRequest interface exposed by browsers. Additional benefits of HttpClient include testability features, typed request and response objects, request and response interception, Observable apis, and streamlined error handling.

Before you can use the HttpClient, you need to import the Angular HttpClientModule. Most apps do so in the root AppModule.

Having imported HttpClientModule into the AppModule, you can inject the HttpClient into an application.

Applications often request JSON data from the server. For example, the app might need a configuration file on the server, config.json, that specifies resource URLs.

The ConfigService fetches this file with a get() method on HttpClient.

What is the difference between a promise and an observable?

<u>Promise</u>

The semantics of Angular dictate that you use promises as a sort of ‘callback handle’ – do something asynchronous in a service, return a promise, and when the asynchronous work is done, the promise’s THEN function is triggered

<u>Observable</u>

Observables provide support for passing messages between publishers and subscribers in your application. Observables offer significant benefits over other techniques for event handling, asynchronous programming, and handling multiple values.

Observables are declarative—that is, you define a function for publishing values, but it is not executed until a consumer subscribes to it. The subscribed consumer then receives notifications until the function completes, or until they unsubscribe.

An observable can deliver multiple values of any type—literals, messages, or events, depending on the context. The API for receiving values is the same whether the values are delivered synchronously or asynchronously. Because setup and teardown logic are both handled by the observable, your application code only needs to worry about subscribing to consume values, and when done, unsubscribing. Whether the stream was keystrokes, an HTTP response, or an interval timer, the interface for listening to values and stopping listening is the same.

Because of these advantages, observables are used extensively within Angular, and are recommended for app development as well.

Hibernate

What is Hibernate?

Hibernate ORM is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database.

What are the benefits of the Hibernate ORM?

Hibernate has a layered architecture which helps the user to operate without having to know the underlying APIs. Hibernate makes use of the database and configuration data to provide persistence services (and persistent objects) to the application.

What are the differences between Hibernate and JDBC?

Like JDBC, Hibernate supports Structured Query Language (SQL) Hibernate Query Language (HQL) is similar to SQL in that it is an object-oriented query language. However, it doesn't operate on tables like SQL but rather uses persistent objects and their properties.

How is Hibernate configured? (steps, please)

Hibernate requires a set of configuration settings related to database and other related parameters. All such information is usually supplied as a standard Java properties file called hibernate.properties, or as an XML file named hibernate.cfg.xml.

Create the Persistent class

Create the mapping file for Persistent class

Create the Configuration file

Create the class that retrieves or stores the persistent object

Load the jar file

Run the first hibernate application by using command prompt

What are the commonly used interfaces of Hibernate?

An Object/relational mappings are usually defined in an XML document.

How do you map a table to a class? (details!)

An Object/relational **mappings** are usually defined in an XML document. ... The **mapping** document is an XML document having <**hibernate-mapping**> as the root element, which contains all the <class> elements. The <class> elements are used to define specific **mappings** from a Java classes to the database tables.

What are the object states of Hibernate?

Transient **-** an object is transient if it has just been instantiated using the new operator, and it is not associated with a Hibernate Session. It has no persistent representation in the database and no identifier value has been assigned. Transient instances will be destroyed by the garbage collector if the application does not hold a reference anymore. Use the Hibernate Session to make an object persistent (and let Hibernate take care of the SQL statements that need to be executed for this transition).

Persistent - a persistent instance has a representation in the database and an identifier value. It might just have been saved or loaded, however, it is by definition in the scope of a Session. Hibernate will detect any changes made to an object in persistent state and synchronize the state with the database when the unit of work completes. Developers do not execute manual UPDATE statements, or DELETE statements when an object should be made transient.

Detached - a detached instance is an object that has been persistent, but its Session has been closed. The reference to the object is still valid, of course, and the detached instance might even be modified in this state. A detached instance can be reattached to a new Session at a later point in time, making it (and all the modifications) persistent again. This feature enables a programming model for long running units of work that require user think-time. We call them application transactions, i.e., a unit of work from the point of view of the user.

What kind of Hibernate exception have you had and how did you fix it?

1. LazyInitializationException

Hibernate throws a LazyInitializationException if you try to access a not initialized relationship to another entity without an active session. You can see a simple example for this in the following code snippet.

```
EntityManager em =
emf.createEntityManager();
                              em.getTransaction().begin();



                              Author a = em.find(Author.class, 1L);



                              em.getTransaction().commit();

                              em.close();


                              log.info(a.getFirstName() + " " + a.getLastName() + "
                              wrote "+a.getBooks().size() + " books.")
```

What is the difference between eager and lazy fetching (not just get vs load...)? What is the default?

One big difference is that EAGER fetch strategy allows to use fetched data object without session. ... However, in case of lazy loading strategy, lazy loading marked object does not retrieve data if session is disconnected (after session.close() statement). All that can be made by hibernate proxy.

What are the properties of a transaction?

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

- **Atomicity** − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
- **Consistency** − The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
- **Durability** − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
- **Isolation** − In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

what is Automatic Dirty Checking?

Hibernate monitors all persistent objects. At the end of a unit of work, it knows which objects have been modified. Then it calls update statement on all updated objects. This process of monitoring and updating only objects that have been changed is called automatic dirty checking in hibernate.

get() vs load()?

Main difference between get() vs load method is that get() involves database hit if object doesn't exists in Session Cache and returns a fully initialized object which may involve several database call while load method can return proxy in place and only initialize the object or hit the database if any method other than getId() is called on persistent or entity object. This lazy initialization can save couple of database round-trip which result in better performance.

update() vs merge()?

Both the MERGE and UPDATE statements are designed to modify data in one table based on data from another, but MERGE can do much more. Whereas UPDATE can only modify column values you can use the MERGE statement to synchronize all data changes such as removal and addition of row.

save() vs persist()?

Main difference between save and saveOrUpdate method is that save() generates a new identifier and INSERT record into database while saveOrUpdate can either INSERT or UPDATE based upon existence of record. Clearly saveOrUpdate is more flexible in terms of use but it involves an extra processing to find out whether record already exists in table or not.

In summary save() method saves records into database by INSERT SQL query, Generates a new identifier and return the Serializable identifier back.

On the other hand saveOrUpdate() method either INSERT or UPDATE based upon existence of object in database. If persistence object already exists in database then UPDATE SQL will execute and if there is no corresponding object in database then INSERT will run.

Other Session methods?

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.

The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed. The main function of the Session is to offer, create, read, and delete operations for instances of mapped entity classes.

Instances may exist in one of the following three states at a given point in time −

- **transient** − A new instance of a persistent class, which is not associated with a Session and has no representation in the database and no identifier value is considered transient by Hibernate.

- **persistent** − You can make a transient instance persistent by associating it with a Session. A persistent instance has a representation in the database, an identifier value and is associated with a Session.
- **detached** − Once we close the Hibernate Session, the persistent instance will become a detached instance.

A Session instance is serializable if its persistent classes are serializable. A typical transaction should use the following idiom −

```
Session session = factory.openSession();
Transaction tx = null;

try {
   tx = session.beginTransaction();
   // do some work
   ...
   tx.commit();
}

catch (Exception e) {
   if (tx!=null) tx.rollback();
   e.printStackTrace();
} finally {
   session.close();
}
```

If the Session throws an exception, the transaction must be rolled back and the session must be discarded.

Session Interface Methods

There are number of methods provided by the Session interface, but I'm going to list down a few important methods only, which we will use in this tutorial. You can check Hibernate documentation for a complete list of methods associated with Session and SessionFactory.

Sr.No.	Session Methods & Description

1	Transaction beginTransaction() Begin a unit of work and return the associated Transaction object.
2	void cancelQuery() Cancel the execution of the current query.
3	void clear() Completely clear the session.
4	Connection close() End the session by releasing the JDBC connection and cleaning up.
5	Criteria createCriteria(Class persistentClass) Create a new Criteria instance, for the given entity class, or a superclass of an entity class.
6	Criteria createCriteria(String entityName) Create a new Criteria instance, for the given entity name.

7	Serializable getIdentifier(Object object) Return the identifier value of the given entity as associated with this session.
8	Query createFilter(Object collection, String queryString) Create a new instance of Query for the given collection and filter string.
9	Query createQuery(String queryString) Create a new instance of Query for the given HQL query string.
10	SQLQuery createSQLQuery(String queryString) Create a new instance of SQLQuery for the given SQL query string.
11	void delete(Object object) Remove a persistent instance from the datastore.
12	void delete(String entityName, Object object) Remove a persistent instance from the datastore.

13	Session get(String entityName, Serializable id) Return the persistent instance of the given named entity with the given identifier, or null if there is no such persistent instance.
14	SessionFactory getSessionFactory() Get the session factory which created this session.
15	void refresh(Object object) Re-read the state of the given instance from the underlying database.
16	Transaction getTransaction() Get the Transaction instance associated with this session.
17	boolean isConnected() Check if the session is currently connected.
18	boolean isDirty() Does this session contain any changes which must be synchronized with the database?

19	boolean isOpen() Check if the session is still open.
20	Serializable save(Object object) Persist the given transient instance, first assigning a generated identifier.
21	void saveOrUpdate(Object object) Either save(Object) or update(Object) the given instance.
22	void update(Object object) Update the persistent instance with the identifier of the given detached instance.
23	void update(String entityName, Object object) Update the persistent instance with the identifier of the given detached instance.

What is the difference between L1 and L2 caching?

L1 is "level-1" cache memory, usually built onto the microprocessor chip itself. For example, the Intel MMX microprocessor comes with 32 thousand bytes of L1.

L2 (that is, level-2) cache memory is on a separate chip (possibly on an expansion card) that can be accessed more quickly than the larger "main" memory.

Is the SessionFactory thread safe? Is the Session?

SessionFactory is Hibernates concept of a single data store and is threadsafe so that many threads can access it concurrently and request for sessions and the immutable cache of compiled mappings for a single database. A SessionFactory is usually only built once at startup.

Session is not threadsafe

What is the difference between Query and CriteriaQuery?

Query

Hibernate Query Language (HQL) is an object-oriented query language, similar to SQL, but instead of operating on tables and columns, HQL works with persistent objects and their properties. HQL queries are translated by Hibernate into conventional SQL queries, which in turns perform action on database.

Criteria Query

The Hibernate Criteria Query Language (HCQL) is used to fetch the records based on the specific criteria. The Criteria interface provides methods to apply criteria such as retreiving all the records of table whose salary is greater than 50000 etc

How and why to create a NamedQuery?

In Hibernate, a named query is a JPQL or SQL expression with predefined unchangeable query string. You can define a named query either in hibernate mapping file or in an entity class.

Welcome to Hibernate Named Query Example Tutorial. We saw how we can use HQL and Native SQL Query in Hibernate. If there are a lot of queries, then they will cause a code mess because all the queries will be scattered throughout the project. That’s why Hibernate provides Named Query that we can define at a central location and use them anywhere in the code. We can created named queries for both HQL and Native SQL.

How and why or why not to use native SQL in Hibernate?

You can use native SQL to express database queries if you want to utilize database-specific features such as query hints or the CONNECT keyword in Oracle. Hibernate

3.x allows you to specify handwritten SQL, including stored procedures, for all create, update, delete, and load operations.
Your application will create a native SQL query from the session with the createSQLQuery() method on the Session interface −

```
public SQLQuery createSQLQuery(String sqlString) throws HibernateException
```

After you pass a string containing the SQL query to the createSQLQuery() method, you can associate the SQL result with either an existing Hibernate entity, a join, or a scalar result using addEntity(), addJoin(), and addScalar() methods respectively.

What is HQL? How is it different from native SQL?

Hibernate Query Language (HQL) is an object-oriented query language, similar to SQL, but instead of operating on tables and columns, HQL works with persistent objects and their properties. HQL queries are translated by Hibernate into conventional SQL queries, which in turns perform action on database.

What is JPA?

The Java Persistence API is a Java application programming interface specification that describes the management of relational data in applications using Java Platform, Standard Edition and Java Platform, Enterprise Edition.

What JPA annotations have you used?

JPA annotations are used in mapping java objects to the database tables, columns etc. Hibernate is the most popular implement of JPA specification and provides some additional annotations.

javax.persistence.Entity: Specifies that the class is an entity. This annotation can be applied on Class, Interface of Enums.

@Table: It specifies the table in the database with which this entity is mapped. In the example below the data will be stores in the “employee” table. Name attribute of @Table annotation is used to specify the table name

@Column: Specify the column mapping using @Column annotation. Name attribute of this annotation is used for specifying the table’s column name.

@Id: This annotation specifies the primary key of the entity.

@GeneratedValue: This annotation specifies the generation strategies for the values of primary keys.

@Version: We can control versioning or concurrency using this annotation.

@OrderBy: Sort your data using @OrderBy annotation. In example below, it will sort all employees_address by their id in ascending order.

@Transient: Every non static and non-transient property of an entity is considered persistent, unless you annotate it as @Transient.

@Lob: Large objects are declared with @Lob.

@OneToOne

@ManyToOne

@ManyToMany

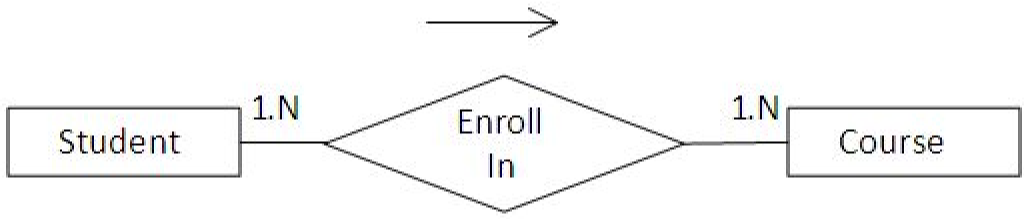
These are used for multiplicity

When can you see a LazyInitializationException?

This error means that you're trying to access a lazily-loaded property or collection, but the hibernate session is closed or not available . Lazy loading in Hibernate means that the object will not be populated (via a database query) until the property/collection is accessed in code. Hibernate accomplishes this by creating a dynamic proxy object that will hit the database only when you first use the object. In order for this to work, your object must be attached to an open Hibernate session throughout its lifecycle.

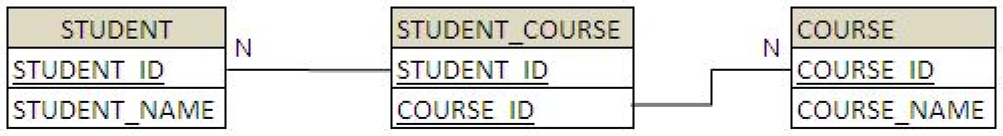
How to represent multiplicity relationships in Hibernate?

In this example you will learn how to map many-to-many relationship using Hibernate Annotations. Consider the following relationship between Student and Course entity.



According to the relationship a student can enroll in any number of courses and the course can have any number of students.

To create this relationship you need to have a STUDENT, COURSE and STUDENT_COURSE table. The relational model is shown below.



To create the STUDENT, COURSE and STUDENT_COURSE table you need to create the following Java Class files.

Student class is used to create the STUDENT and STUDENT_COURSE table.

```
        package com.vaannila.student;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "STUDENT")
```

```
public class Student {

        private long studentId;
        private String studentName;
        private Set<Course> courses = new HashSet<Course>(0);

        public Student() {
        }

        public Student(String studentName) {
                this.studentName = studentName;
        }

        public Student(String studentName, Set<Course> courses) {
                this.studentName = studentName;
                this.courses = courses;
        }

        @Id
        @GeneratedValue
        @Column(name = "STUDENT_ID")
        public long getStudentId() {
                return this.studentId;
        }

        public void setStudentId(long studentId) {
                this.studentId = studentId;
        }

        @Column(name = "STUDENT_NAME", nullable = false, length = 100)
        public String getStudentName() {
                return this.studentName;
        }

        public void setStudentName(String studentName) {
                this.studentName = studentName;
        }
```

```
	@ManyToMany(cascade = CascadeType.ALL)
	@JoinTable(name = "STUDENT_COURSE", joinColumns = { @JoinColumn(name =
"STUDENT_ID") }, inverseJoinColumns = { @JoinColumn(name = "COURSE_ID") })
	public Set<Course> getCourses() {
		return this.courses;
	}

	public void setCourses(Set<Course> courses) {
		this.courses = courses;
	}

}
```

The @ManyToMany annotation is used to create the many-to-many relationship between the Student and Course entities. The @JoinTable annotation is used to create the STUDENT_COURSE link table and @JoinColumn annotation is used to refer the linking columns in both the tables.

Course class is used to create the COURSE table.

```
	package com.vaannila.student;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="COURSE")
public class Course {

	private long courseId;
	private String courseName;

	public Course() {
	}
	public Course(String courseName) {
		this.courseName = courseName;
	}

	@Id
	@GeneratedValue
	@Column(name="COURSE_ID")
	public long getCourseId() {
		return this.courseId;
	}

	public void setCourseId(long courseId) {
		this.courseId = courseId;
	}

	@Column(name="COURSE_NAME", nullable=false)
	public String getCourseName() {
		return this.courseName;
	}

	public void setCourseName(String courseName) {
		this.courseName = courseName;
	}

}
```

Now create the hibernate configuration file with the Student and Course class mapping.

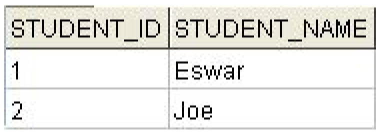
```
	<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
		"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
		"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class"> org.hsqldb.jdbcDriver</property>
        <property name="hibernate.connection.url"> jdbc:hsqldb:hsql://localhost</property>
        <property name="hibernate.connection.username">sa</property>
        <property name="connection.password"></property>
        <property name="connection.pool_size">1</property>
        <property name="hibernate.dialect"> org.hibernate.dialect.HSQLDialect</property>
        <property name="show_sql">true</property>
        <property name="hbm2ddl.auto">create-drop</property>
        <mapping class="com.vaannila.student.Student" />
        <mapping class="com.vaannila.student.Course" />
    </session-factory>
</hibernate-configuration>
```

Create the Main class to run the example.

```
package com.vaannila.student;

import java.util.HashSet;
import java.util.Set;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.Transaction;

import com.vaannila.util.HibernateUtil;

public class Main {

        public static void main(String[] args) {

                Session session = HibernateUtil.getSessionFactory().openSession();
                Transaction transaction = null;
                try {
                        transaction = session.beginTransaction();

                        Set<Course> courses = new HashSet<Course>();
                        courses.add(new Course("Maths"));
                        courses.add(new Course("Computer Science"));

                        Student student1 = new Student("Eswar", courses);
                        Student student2 = new Student("Joe", courses);
                        session.save(student1);
                        session.save(student2);

                        transaction.commit();
                } catch (HibernateException e) {
                        transaction.rollback();
                        e.printStackTrace();
                } finally {
                        session.close();
                }

        }
}
```

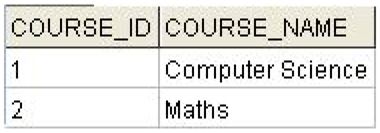
On executing the Main class you will see the following output.

The STUDENT table has two records.



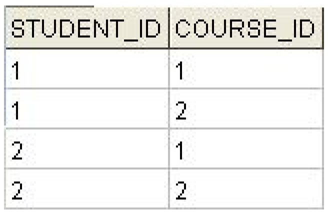
STUDENT_ID	STUDENT_NAME
1	Eswar
2	Joe

The *COURSE* table has two records.



COURSE_ID	COURSE_NAME
1	Computer Science
2	Maths

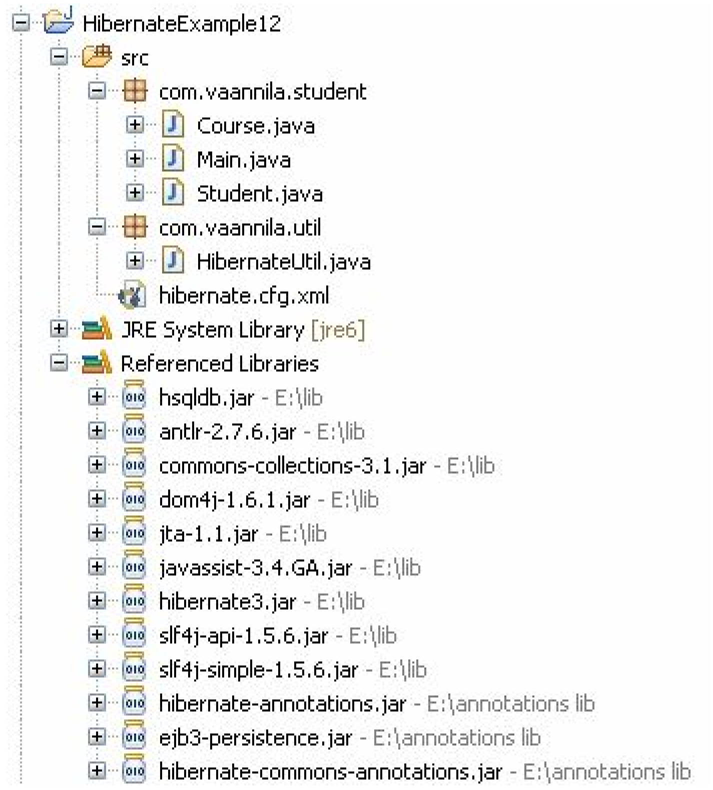
The *STUDENT_COURSE* table has four records to link the student and courses.



STUDENT_ID	COURSE_ID
1	1
1	2
2	1
2	2

Each student has enrolled in the same two courses, this illustrates the many-to-many mapping.

The folder structure of the example is shown below.



```
HibernateExample12
  src
    com.vaannila.student
      Course.java
      Main.java
      Student.java
    com.vaannila.util
      HibernateUtil.java
    hibernate.cfg.xml
  JRE System Library [jre6]
  Referenced Libraries
    hsqldb.jar - E:\lib
    antlr-2.7.6.jar - E:\lib
    commons-collections-3.1.jar - E:\lib
    dom4j-1.6.1.jar - E:\lib
    jta-1.1.jar - E:\lib
    javassist-3.4.GA.jar - E:\lib
    hibernate3.jar - E:\lib
    slf4j-api-1.5.6.jar - E:\lib
    slf4j-simple-1.5.6.jar - E:\lib
    hibernate-annotations.jar - E:\annotations lib
    ejb3-persistence.jar - E:\annotations lib
    hibernate-commons-annotations.jar - E:\annotations lib
```

What is a Hibernate proxy object?

A proxy is a subclass implemented at runtime. Hibernate creates a proxy (a subclass of the class being fetched) instead of querying the database directly, and this proxy will load the "real" object from the database whenever one of its methods is called.

Spring

What is your experience with the Spring framework?

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.

How did you implement Spring on your last project?

We used SpringBoot to automatically map beans to CRUD methods, implemented Spring Security to authenticate users and store salts and hashes.

What modules have you worked with?

Core, Beans, ORM, MVC, AOP

What is dependency injection, and how is it related to inversion of control?

IOC (Inversion of control) is a general parent term while DI (Dependency injection) is a subset of IOC. IOC is a concept where the flow of application is inverted. DI provides objects that an object needs. So rather than the dependencies construct themselves they are injected by some external means.

What is the benefit of DI?

There are several benefits from using dependency injection containers rather than having components satisfy their own dependencies. Some of these benefits are:

- Reduced Dependencies
- Reduced Dependency Carrying
- More Reusable Code
- More Testable Code
- More Readable Code

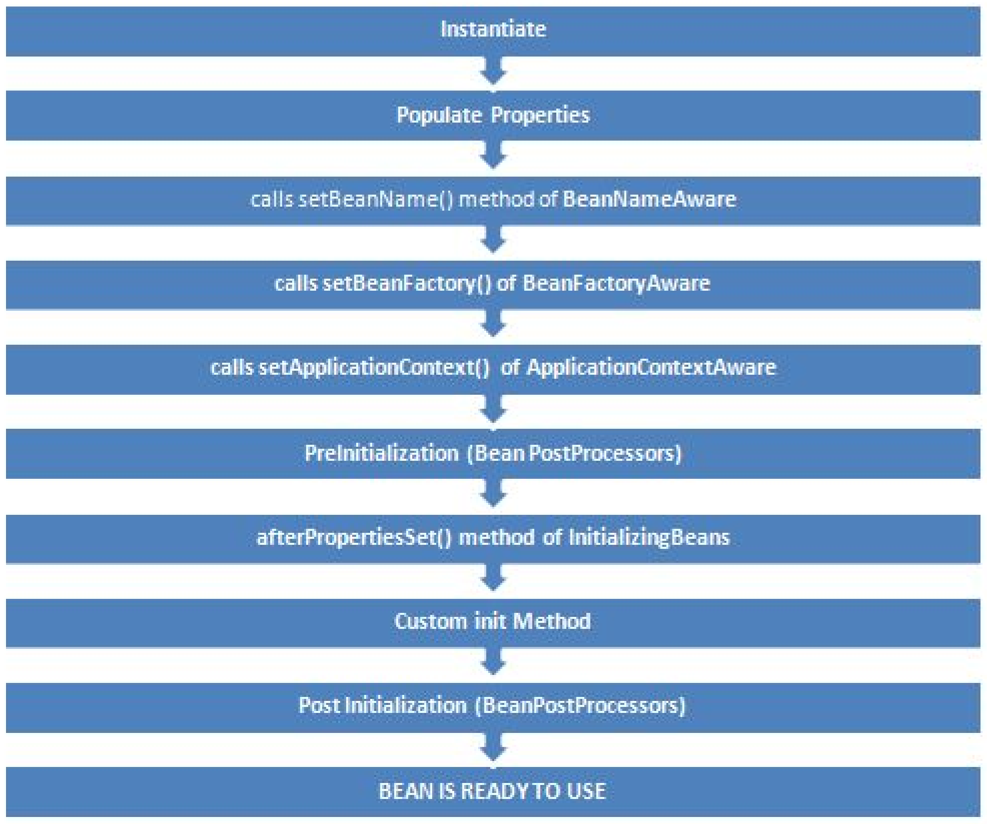
What is the Application Context?

The ApplicationContext is the central interface within a Spring application for providing configuration information to the application. It is read-only at run time, but can be reloaded if necessary and supported by the application. A number of classes implement the ApplicationContext interface, allowing for a variety of configuration options and types of applications.

What implementations of the application context have you used?

- Bean factory methods for accessing application components.
- The ability to load file resources in a generic fashion.
- The ability to publish events to registered listeners.
- The ability to resolve messages to support internationalization.
- Inheritance from a parent context.

What is the lifecycle of a Spring bean? How much of it is handled by Spring, and how much can we write ourselves? Why would we want to?


Instantiate
Populate Properties
calls setBeanName() method of BeanNameAware
calls setBeanFactory() of BeanFactoryAware
calls setApplicationContext() of ApplicationContextAware
PreInitialization (Bean PostProcessors)
afterPropertiesSet() method of InitializingBeans
Custom init Method
Post Initialization (BeanPostProcessors)
BEAN IS READY TO USE

How can we specify custom init, destroy methods for one bean? For all of the beans?

When you write initialization and destroy method callbacks that do not use the Spring-specific InitializingBean and DisposableBean callback interfaces, you typically write methods with names such as init(), initialize(), dispose(), and so on. Ideally, the names of such lifecycle callback methods are standardized across a project so that all developers use the same method names and ensure consistency.

You can configure the Spring container to look for named initialization and destroy callback method names on *every* bean. This means that you, as an application developer, can write your application classes and use an initialization callback called init(), without having to configure an init-method="init" attribute with each bean definition. The Spring IoC container calls that method when the bean is created (and in accordance with the standard lifecycle callback contract described previously). This feature also enforces a consistent naming convention for initialization and destroy method callbacks.

Suppose that your initialization callback methods are named init() and destroy callback methods are named destroy(). Your class will resemble the class in the following example.

```
public class DefaultBlogService implements BlogService {

    private BlogDao blogDao;

    public void setBlogDao(BlogDao blogDao) {
        this.blogDao = blogDao;
    }

    // this is (unsurprisingly) the initialization callback method
    public void init() {
        if (this.blogDao == null) {
            throw new IllegalStateException("The [blogDao] property must be set.");
        }
    }
}
<beans default-init-method="init">

    <bean id="blogService" class="com.foo.DefaultBlogService">
        <property name="blogDao" ref="blogDao" />
    </bean>

</beans>
```

The presence of the default-init-method attribute on the top-level <beans/> element attribute causes the Spring IoC container to recognize a method called init on beans as the initialization method callback. When a bean is created and assembled, if the beans class has such a method, it is invoked at the appropriate time.

You configure destroy method callbacks similarly (in XML, that is) by using the default-destroy-method attribute on the top-level <beans/> element.
Where existing bean classes already have callback methods that are named at variance with the convention, you can override the default by specifying (in XML, that is) the method name using the init-method and destroy-method attributes on the <bean/> itself.

The Spring container guarantees that a configured initialization callback is called immediately after a bean is supplied with all dependencies. Thus the initialization callback is called on the raw bean reference, which means that AOP interceptors and so forth are not yet applied to the bean. A target bean is fully created *first*, *then* an AOP proxy (for example) with its interceptor chain is applied. If the target bean and the proxy are defined separately, your code can even interact with the raw target bean, bypassing the proxy. Hence, it would be inconsistent to apply the interceptors to the init method, because doing so would couple the lifecycle of the target bean with its proxy/interceptors and leave strange semantics when your code interacts directly to the raw target bean.

What is bean wiring? How is this different from autowiring?

Bean wiring means creating associations between application components i.e. beans within the spring container.

Autowiring using property name. Spring container looks the properties of the beans on which autowire attribute is set to byName in the XML configuration file. It tries to match and wire its properties with the beans defined by the same name in the configuration file.

Dependency injection types?

Spring supports 2 types of dependency injection, they are:

1) Constructor-based dependency injection: It is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class.

2) Setter-based dependency injection: It is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

What functionality does Spring Core provide?

The Spring Framework includes several modules that provide a range of services:
Spring Core Container: this is the base module of Spring and provides spring containers (BeanFactory and ApplicationContext).

How is it set up? Why is it useful?

Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, and reusable code.

Spring is lightweight when it comes to size and transparency. The basic version of Spring framework is around 2MB.

The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promotes good programming practices by enabling a POJO-based programming model.

Benefits of Using the Spring Framework

719-284-5940
47424
Following is the list of few of the great benefits of using Spring Framework −

- Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product.
- Spring is organized in a modular fashion. Even though the number of packages and classes are substantial, you have to worry only about the ones you need and ignore the rest.
- Spring does not reinvent the wheel, instead it truly makes use of some of the existing technologies like several ORM frameworks, logging frameworks, JEE, Quartz and JDK timers, and other view technologies.
- Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Furthermore, by using JavaBeanstyle POJOs, it becomes easier to use dependency injection for injecting test data.
- Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks such as Struts or other over-engineered or less popular web frameworks.
- Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.
- Lightweight IoC containers tend to be lightweight, especially when compared to EJB containers, for example. This is beneficial for developing and deploying applications on computers with limited memory and CPU resources.
- Spring provides a consistent transaction management interface that can scale down to a local transaction (using a single database, for example) and scale up to global transactions (using JTA, for example).

You can create a new Spring Project directly through your IDE (Eclipse/STS) or use Spring Boot's online initializer https://start.spring.io/

What is @Component?

Spring Component annotation is used to denote a class as Component. It means that Spring framework will autodetect these classes for dependency injection when annotation-based configuration and classpath scanning is used.

Under what circumstances would I want to use singleton or prototype scope?

Prototype scope = A new object is created each time it is injected/looked up. It will use new SomeClass() each time.

Singleton scope = (Default) The same object is returned each time it is injected/looked up. Here it will instantiate one instance of SomeClass and then return it each time.

How could I use Spring AOP to log the stack trace of an exception to an external file?

1. Add support for spring-aop and aspectj
If you are using maven, and not using spring-aop already, just add this two new dependencies.

```
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-aop</artifactId>
<version>${org.springframework.version}</version>
</dependency>

<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjtools</artifactId>
<version>${org.aspectj.verision}</version>
</dependency>
```

2. Define your aspect and pointcuts

There are two ways two define aspects : With @Aspect annotation, or in Spring`s application.xml file. Here we`ll be using Spring annotation.

Just Create aspect class file as below. Also, to enable automatic generation of proxies, tag “<aop:aspectj-autoproxy/>”Â needs to be added in applicationContext.xml file.

```
@Component
@Aspect
public class MyAspect {

  private final Log log = LogFactory.getLog(this.getClass());

  @Around("execution(* com.foo.bar..*.*(..))")
  public Object logTimeMethod(ProceedingJoinPoint joinPoint) throws
Throwable {

                StopWatch stopWatch = new StopWatch();
                stopWatch.start();

                Object retVal = joinPoint.proceed();

                stopWatch.stop();

                StringBuffer logMessage = new StringBuffer();

logMessage.append(joinPoint.getTarget().getClass().getName());
                logMessage.append(".");
                logMessage.append(joinPoint.getSignature().getName());
```

```
            logMessage.append("(");
            // append args
            Object[] args = joinPoint.getArgs();
            for (int i = 0; i < args.length; i++) {
                 logMessage.append(args[i]).append(",");
            }
            if (args.length > 0) {
                 logMessage.deleteCharAt(logMessage.length() - 1);
            }

            logMessage.append(")");
            logMessage.append(" execution time: ");
            logMessage.append(stopWatch.getTotalTimeMillis());
            logMessage.append(" ms");
            log.info(logMessage.toString());
            return retVal;
    }

}
```

In this example, method calls are logged with execution time.. For example if you want to log what method is returning, use @AfterReturnung Advice:

```
@AfterReturning(pointcut = "execution(* com.foo.bar..*.*(..))", returning
= "retVal")
public void logAfterMethod(JoinPoint joinPoint, Object retVal) {
...
}
```

If you want to dynamically enable/disable logging, one way this can be done is introducing new flag in Logging Aspect. Than just add new Join point that will intercept your’s specific calls of changing this flag.

You can also completely disable aspects with just commenting Pointcut definition, or you can externalize this configuration using Spring.

How can I determine which methods this aspect will be applied to?

Aspect-Oriented Programming entails breaking down program logic into distinct parts called so-called concerns. The functions that span multiple points of an application are called cross-cutting concerns and these cross-cutting concerns are conceptually separate from the application's business logic. There are various common good examples of aspects like logging, auditing, declarative transactions, security, caching, etc.

The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. Dependency Injection helps you decouple your application objects from each other and AOP helps you decouple cross-cutting concerns from the objects that they affect. AOP is like triggers in programming languages such as Perl, .NET, Java, and others.

Spring AOP module provides interceptors to intercept an application. For example, when a method is executed, you can add extra functionality before or after the method execution.

Types of Advice

Spring aspects can work with five kinds of advice mentioned as follows −

Sr.No	Advice & Description
1	before Run advice before the a method execution.

2	after Run advice after the method execution, regardless of its outcome.
3	after-returning Run advice after the a method execution only if method completes successfully.
4	after-throwing Run advice after the a method execution only if method exits by throwing an exception.
5	around Run advice before and after the advised method is invoked.

What else can Spring AOP be used for?

auditing, declarative transactions, security, caching

What is autoproxying?

It implements the BeanPostProcessor interface, which in its implementation replaces the bean (target) with a proxy.

What is Spring MVC?

The Spring Web MVC framework provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications. The

MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providi/8 ng a loose coupling between these elements.

- The Model encapsulates the application data and in general they will consist of POJO.
- The View is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
- The Controller is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.

How is it set up? Why is it useful?

In Spring's approach to building web sites, HTTP requests are handled by a controller. You can easily identify these requests by the `@Controller` annotation. In the following example, the GreetingController handles GET requests for /greeting by returning the name of a `View`, in this case, "greeting". A `View` is responsible for rendering the HTML content:

`src/main/java/hello/GreetingController.java`

```java
package hello;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class GreetingController {

    @GetMapping("/greeting")
    public String greeting(@RequestParam(name="name", required=false,
defaultValue="World") String name, Model model) {
        model.addAttribute("name", name);
        return "greeting";
    }
}
```

This controller is concise and simple, but there's plenty going on. Let's break it down step by step.

The `@GetMapping` annotation ensures that HTTP GET requests to `/greeting` are mapped to the `greeting()` method.

`@RequestParam` binds the value of the query String parameter `name` into the `name` parameter of the `greeting()` method. This query String parameter is not `required`; if it is absent in the request, the `defaultValue` of "World" is used. The value of the `name` parameter is added to a `Model` object, ultimately making it accessible to the view template.

The implementation of the method body relies on a view technology, in this case Thymeleaf, to perform server-side rendering of the HTML. Thymeleaf parses the `greeting.html` template below and evaluates the `th:text` expression to render the value of the `${name}` parameter that was set in the controller.

`src/main/resources/templates/greeting.html`

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Getting Started: Serving Web Content</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <p th:text="'Hello, ' + ${name} + '!'" />
</body>
</html>
```

Developing web apps

A common feature of developing web apps is coding a change, restarting your app, and refreshing the browser to view the change. This entire process can eat up a lot of time. To speed up the cycle of things, Spring Boot comes with a handy module known as spring-boot-devtools.

- Enable hot swapping
- Switches template engines to disable caching
- Enables LiveReload to refresh browser automatically
- Other reasonable defaults based on development instead of production

Make the application executable

Although it is possible to package this service as a traditional WAR file for deployment to an external application server, the simpler approach demonstrated below creates a standalone application. You package everything in a single, executable JAR file, driven by a good old Java `main()` method. Along the way, you use Spring's support for embedding the Tomcat servlet container as the HTTP runtime, instead of deploying to an external instance.

`src/main/java/hello/Application.java`

```java
package hello;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

`@SpringBootApplication` is a convenience annotation that adds all of the following:

- `@Configuration` tags the class as a source of bean definitions for the application context.
- `@EnableAutoConfiguration` tells Spring Boot to start adding beans based on classpath settings, other beans, and various property settings.
- Normally you would add `@EnableWebMvc` for a Spring MVC app, but Spring Boot adds it automatically when it sees spring-webmvc on the classpath. This flags the application as a web application and activates key behaviors such as setting up a `DispatcherServlet`.
- `@ComponentScan` tells Spring to look for other components, configurations, and services in the `hello` package, allowing it to find the controllers.

The `main()` method uses Spring Boot's `SpringApplication.run()` method to launch an application. Did you notice that there wasn't a single line of XML? No web.xml file either. This web application is 100% pure Java and you didn't have to deal with configuring any plumbing or infrastructure.

Build an executable JAR

You can run the application from the command line with Gradle or Maven. Or you can build a single executable JAR file that contains all the necessary dependencies, classes, and resources, and run that. This makes it easy to ship, version, and deploy the service as an application throughout the development lifecycle, across different environments, and so forth.

If you are using Gradle, you can run the application using `./gradlew bootRun`. Or you can build the JAR file using `./gradlew build`. Then you can run the JAR file:

```
java -jar build/libs/gs-serving-web-content-0.1.0.jar
```

If you are using Maven, you can run the application using `./mvnw spring-boot:run`. Or you can build the JAR file with `./mvnw clean package`. Then you can run the JAR file:

```
java -jar target/gs-serving-web-content-0.1.0.jar
```

> The procedure above will create a runnable JAR. You can also opt to build a classic WAR fileinstead.

Logging output is displayed. The app should be up and running within a few seconds.

Test the App

Now that the web site is running, visit http://localhost:8080/greeting, where you see:

```
"Hello, World!"
```

Provide a `name` query string parameter with http://localhost:8080/greeting?name=User. Notice how the message changes from "Hello, World!" to "Hello, User!":

```
"Hello, User!"
```

This change demonstrates that the `@RequestParam` arrangement in `GreetingController` is working as expected. The `name` parameter has been given a default value of "World", but can always be explicitly overridden through the query string.

Add a Homepage

Static resources, like HTML or JavaScript or CSS, can easily be served from your Spring Boot application just be dropping them into the right place in the source code. By default Spring Boot serves static content from resources in the classpath at "/static" (or "/public").

The `index.html` resource is special because it is used as a "welcome page" if it exists, which means it will be served up as the root resource, i.e. at `http://localhost:8080/` in our example. So create this file:

`src/main/resources/static/index.html`

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Getting Started: Serving Web Content</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <p>Get your greeting <a href="/greeting">here</a></p>
</body>
</html>
```

and when you restart the app you will see the HTML at http://localhost:8080/.

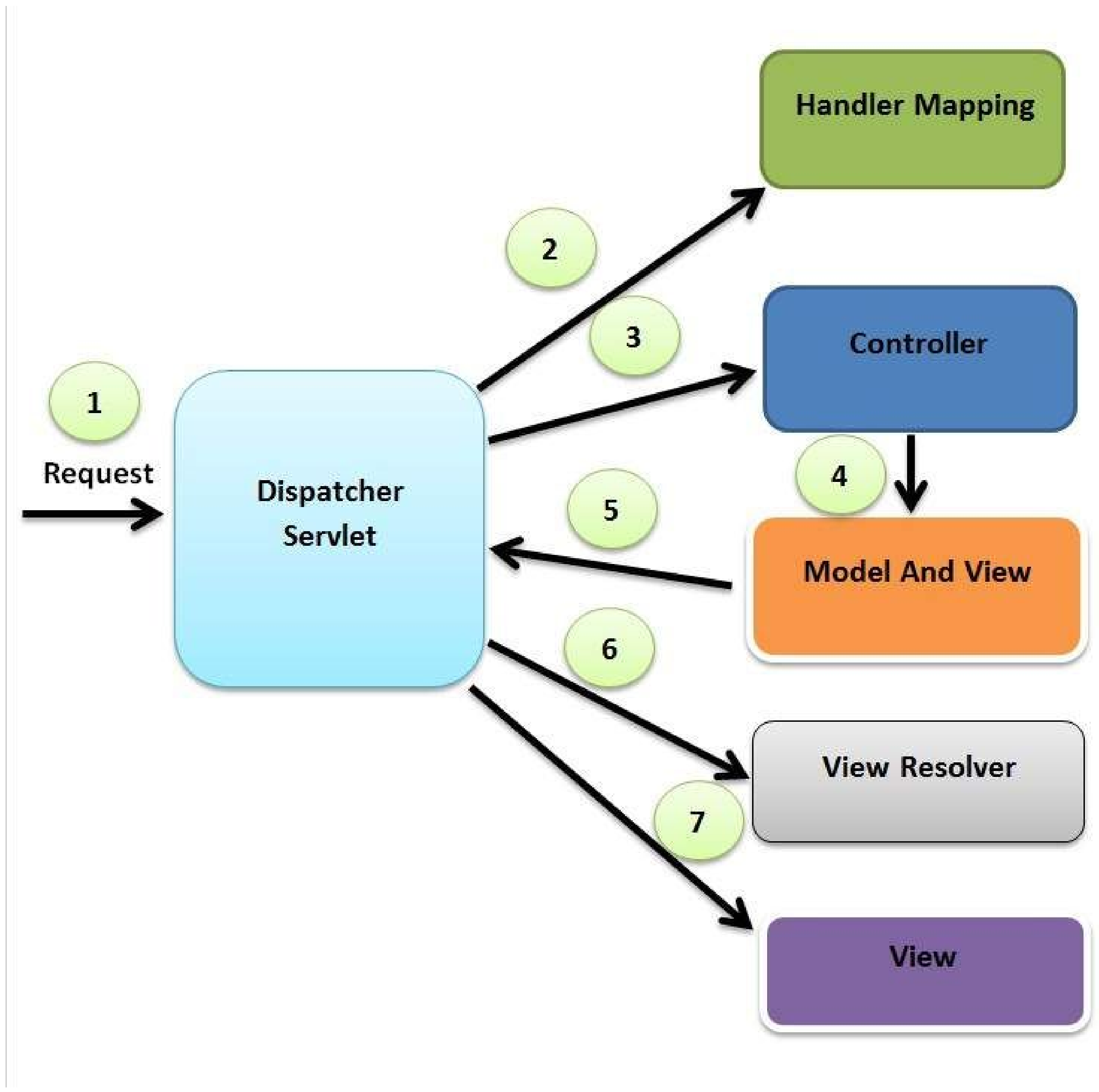
Summary

Congratulations! You have just developed a web page using Spring.

How does Spring MVC implement the Front Controller design pattern?

A front controller is defined as “a controller which handles all requests for a Web Application.” DispatcherServlet (actually a servlet) is the front controller in Spring MVC that intercepts every request and then dispatches/forwards requests to an appropriate controller.

What is the flow of information within a Spring MVC application?



What implementation of the application context is used in Spring MVC?

If I wanted to switch my front-end pages from pure HTML to JSP files, how would I reconfigure my Spring MVC application?

The following example shows how to write a simple web-based application using Spring MVC Framework, which can access static pages along with dynamic pages with the help of <mvc:resources> tag. To start with, let us have a working Eclipse IDE in place and take the following steps to develop a Dynamic Form based Web Application using Spring Web Framework –

Step	Description
1	Create a *Dynamic Web Project* with a name *HelloWeb* and create a package *com.tutorialspoint* under the *src* folder in the created project.
2	Drag and drop below mentioned Spring and other libraries into the folder *WebContent/WEB-INF/lib*.
3	Create a Java class *WebController* under the *com.tutorialspoint* package.
4	Create Spring configuration files *Web.xml* and *HelloWeb-servlet.xml* under the *WebContent/WEB-INF* folder.
5	Create a sub-folder with a name *jsp* under the *WebContent/WEB-INF* folder. Create a view file *index.jsp* under this sub-folder.
6	Create a sub-folder with a name *pages* under the *WebContent/WEB-INF* folder. Create a static file *final.htm* under this sub-folder.
7	The final step is to create the content of all the source and configuration files and export the application as explained below.

Here is the content of WebController.java file

```
package com.tutorialspoint;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
```

```java
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class WebController {
   @RequestMapping(value = "/index", method = RequestMethod.GET)
   public String index() {
      return "index";
   }
   @RequestMapping(value = "/staticPage", method = RequestMethod.GET)
   public String redirect() {
      return "redirect:/pages/final.htm";
   }
}
```

Following is the content of Spring Web configuration file web.xml

```xml
<web-app id = "WebApp_ID" version = "2.4"
   xmlns = "http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation = "http://java.sun.com/xml/ns/j2ee
   http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

   <display-name>Spring Page Redirection</display-name>

   <servlet>
      <servlet-name>HelloWeb</servlet-name>
      <servlet-class>
         org.springframework.web.servlet.DispatcherServlet
      </servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
      <servlet-name>HelloWeb</servlet-name>
      <url-pattern>/</url-pattern>
   </servlet-mapping>

</web-app>
```

Following is the content of another Spring Web configuration file HelloWeb-servlet.xml

```xml
<?xml version = "1.0" encoding = "UTF-8"?>
<beans xmlns = "http://www.springframework.org/schema/beans"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xmlns:context = "http://www.springframework.org/schema/context"
   xmlns:mvc = "http://www.springframework.org/schema/mvc"
   xsi:schemaLocation = "http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
   http://www.springframework.org/schema/mvc
   http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
   http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd"
>

   <context:component-scan base-package="com.tutorialspoint" />

   <bean id = "viewResolver"
      class =
"org.springframework.web.servlet.view.InternalResourceViewResolver">

      <property name = "prefix" value = "/WEB-INF/jsp/" />
      <property name = "suffix" value = ".jsp" />
   </bean>

   <mvc:resources mapping = "/pages/**" location = "/WEB-INF/pages/"
/>
   <mvc:annotation-driven/>

</beans>
```

Here <mvc:resources..../> is being used to map static pages. The mappings attribute must be an Ant pattern that specifies the URL pattern of an http requests. The location attribute must specify one or more valid resource directory locations having static pages including images, stylesheets, JavaScript, and other static content. Multiple resource locations may be specified using a comma-separated list of values.

Following is the content of Spring view file WEB-INF/jsp/index.jsp. This will be a landing page; this page will send a request to access staticPage service method, which will redirect this request to a static page available in WEB-INF/pages folder.

```
<%@taglib uri = "http://www.springframework.org/tags/form" prefix =
"form"%>
<html>
```

```
   <head>
      <title>Spring Landing Page</title>
   </head>

   <body>
      <h2>Spring Landing Page</h2>
      <p>Click below button to get a simple HTML page</p>

      <form:form method = "GET" action = "/HelloWeb/staticPage">
         <table>
            <tr>
               <td>
                  <input type = "submit" value = "Get HTML Page"/>
               </td>
            </tr>
         </table>
      </form:form>
   </body>

</html>
```

Following is the content of Spring view file WEB-INF/pages/final.htm.

```
<html>
   <head>
      <title>Spring Static Page</title>
   </head>

   <body>
      <h2>A simple HTML page</h2>
   </body>
</html>
```

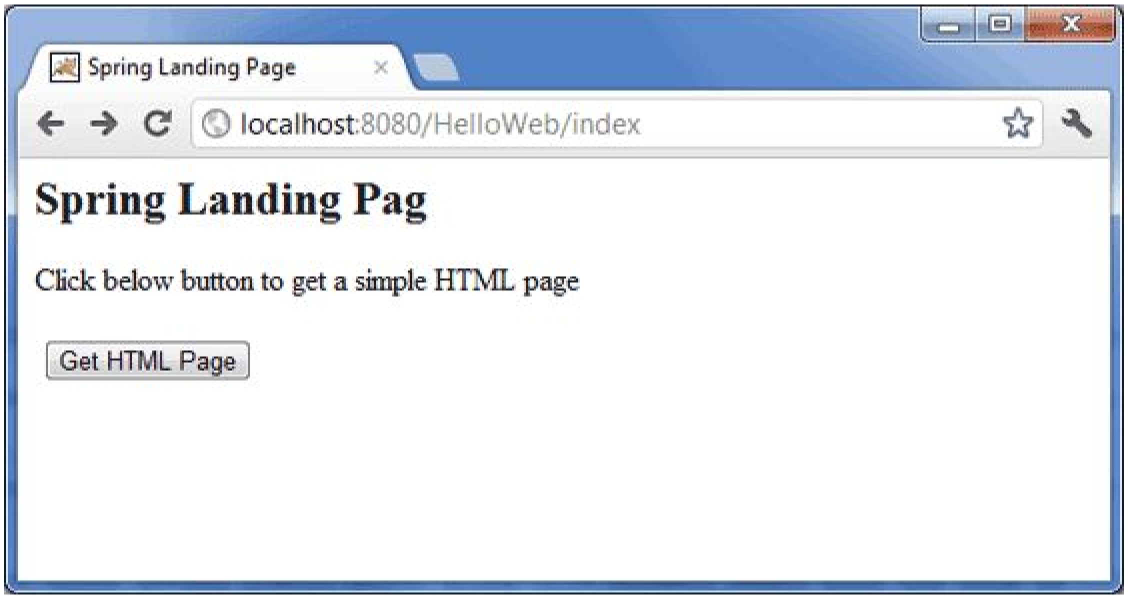
Finally, following is the list of Spring and other libraries to be included in your web application. You simply drag these files and drop them in WebContent/WEB-INF/lib folder.

- commons-logging-x.y.z.jar
- org.springframework.asm-x.y.z.jar
- org.springframework.beans-x.y.z.jar
- org.springframework.context-x.y.z.jar

- org.springframework.core-x.y.z.jar
- org.springframework.expression-x.y.z.jar
- org.springframework.web.servlet-x.y.z.jar
- org.springframework.web-x.y.z.jar
- spring-web.jar

Once you are done creating the source and configuration files, export your application. Right-click on your application and use Export > WAR File option and save your HelloWeb.war file in Tomcat's *webapps* folder.

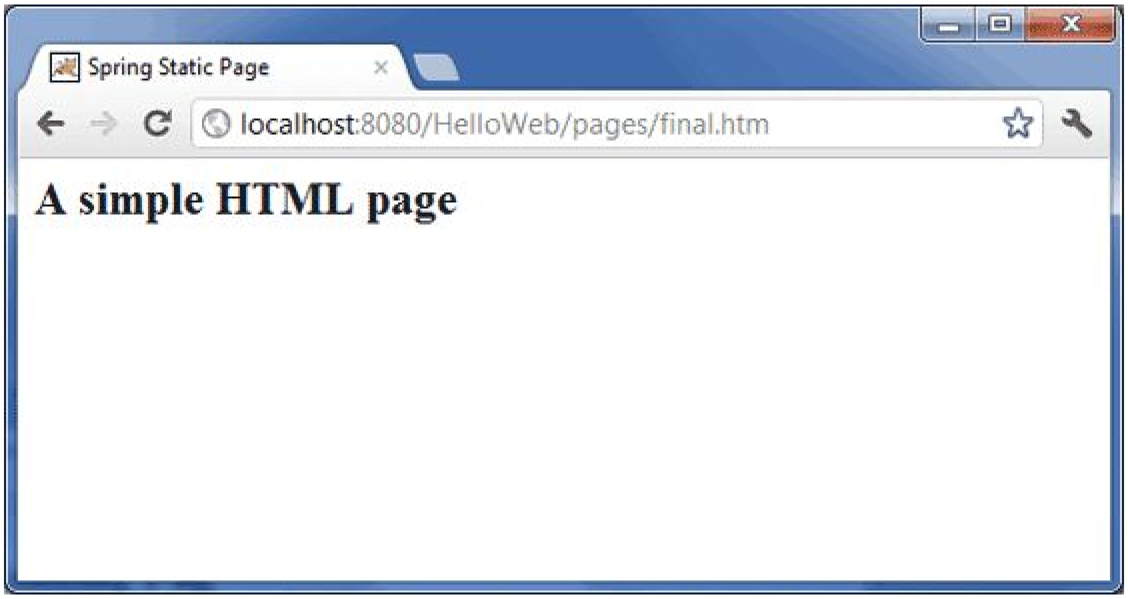
Now start your Tomcat server and make sure you are able to access other web pages from webapps folder using a standard browser. Now try to access the URL http://localhost:8080/HelloWeb/index. If everything is fine with your Spring Web Application, you should see the following result –



Spring Landing Pag

Click below button to get a simple HTML page

Click the "Get HTML Page" button to access a static page mentioned in staticPage service method. If everything is fine with your Spring Web Application, you should see the following result.



A simple HTML page

What is a Spring stereotype?

Annotations denoting the roles of types or methods in the overall architecture (at a conceptual, rather than implementation, level).

The noun definition of stereotype from Merriam-Webster says this:

something conforming to a fixed or general pattern; especially : a standardized mental picture that is held in common by members of a group and that represents an oversimplified opinion, prejudiced attitude, or uncritical judgment

It seems that it is for suggesting a role of particular class that is being annotated. This seems to make sense because it is often recommended that you annotate your Controller classes with @Controller, Service classes with @Service, and so on.

In addition to the obvious component-scanning functionality, Spring suggests that they make nice point-cut demarcations for your AOP needs.

What is the relationship between Spring MVC and Spring REST?

In spring MVC the binding of controller and view is quite strong. It provides a server side view through DispatcherServlet. It is this servlet which defines and sends model to a particular controller and then model and view comes again to this servlet. In this if we see flow then request comes from client, which goes to dispatcher servlet, this servlet with the help of handlerMapping sends it to the particular controller. Then this controller send model and view to dispatcher servlet again after that this servlet sends the model to the particular view name it received. In that view which is server page response is created.

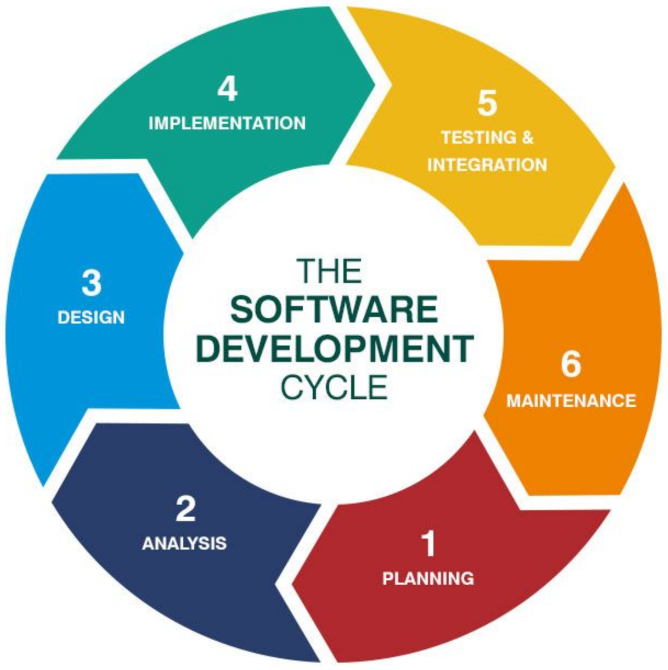
In Spring Rest it also works on MVC architecture only. But in this RestController sets Model object directly into Http response. There it is converted to JSON/XML automatically. So if we define flow here then request comes from client which goes dispatcherServlet, then the dispatcherServlet with the use of handlerMapping send it to particular controller, then this controller sets Model in Http response with the help HttpMessageConverters and this response is sent to client. So we can clearly see that in Rest API direct response was created.

DevOps

What's the difference between SDLC and DevOps?

SDLC

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.



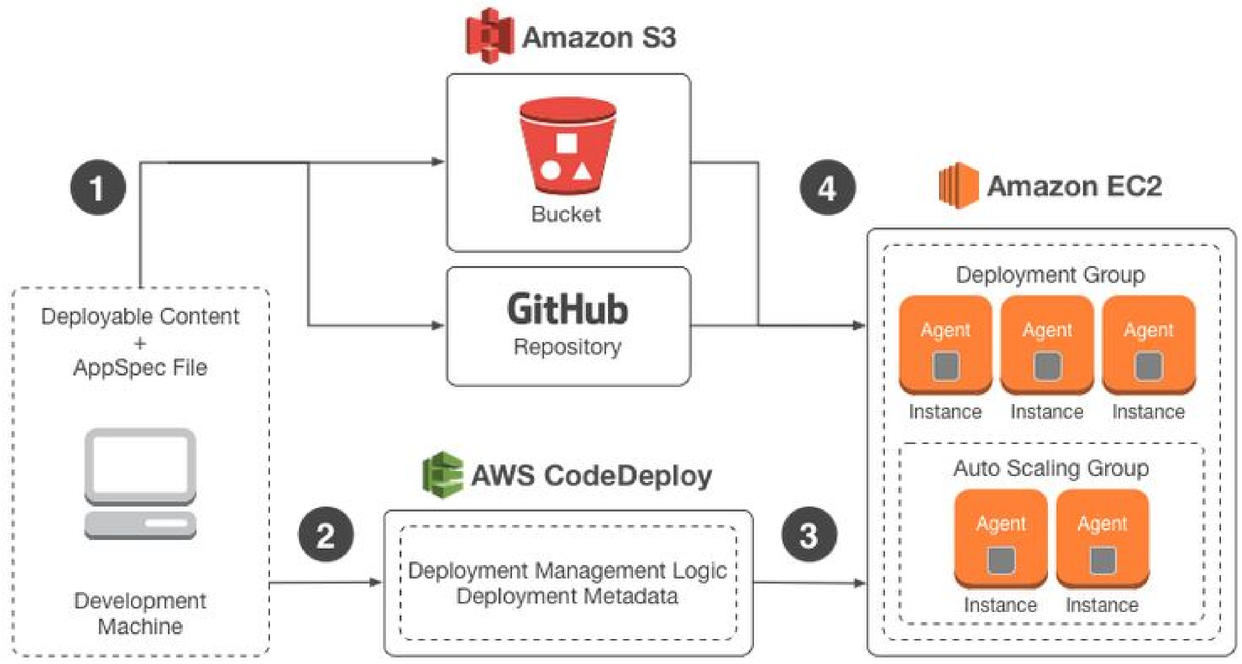
DevOps

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

What’s the difference?

Whereas **SDLC** is mostly concerned with the process of writing software, **DevOps** bridges the gap between software creation and its use, with particular focus on the steps to get software built and deployed. Ideas like continuous integration and deployment or release management are generally considered "**DevOps**" ideas.

Describe the flow of code to deployment in the pipeline.



What is Jenkins responsible for? Where does Maven and Sonarqube fit into this?

Jenkins is an open-source continuous integration software tool written in the Java programming language for testing and reporting on isolated changes in a larger code base in real time. The software enables developers to find and solve defects in a code base rapidly and to automate testing of their builds.

SonarQube is a free and open source code quality measuring and management tool which is developed using java and maintained by sonarsource. It can analyze source code in 20+ different languages. Input can be the project source code or compiled code depending on the language.

Install SonarQube 5.6.6 (LTS *)

```
cd /home

wget https://sonarsource.bintray.com/Distribution/sonarqube/sonarqube-5.6.6.zip

unzip sonarqube-5.6.6.zip

sudo mv sonarqube-5.6.6 /etc/sonarqube
```

Start sonarQube

```
cd  /etc/sonarqube/bin/linux-x86-64
./sonar.sh start
```

Install maven

Run this command to install the latest Apache Maven.

```
sudo apt-get install maven
```

Ensure that you successfully installed maven on your machine

```
mvn -version
Apache Maven 3.3.9 Maven home: /usr/share/maven
Java version: 1.8.0_121, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-8-oracle/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.4.0-75-generic", arch: "amd64", family: "unix"
```

Configure sonarQube with maven

Open settings.xml maven config

```
cd /usr/share/maven
sudo vi settings.xml
```

Go to **pluginGroups** tag and add

```
<pluginGroup>org.sonarsource.scanner.maven</pluginGroup>
```

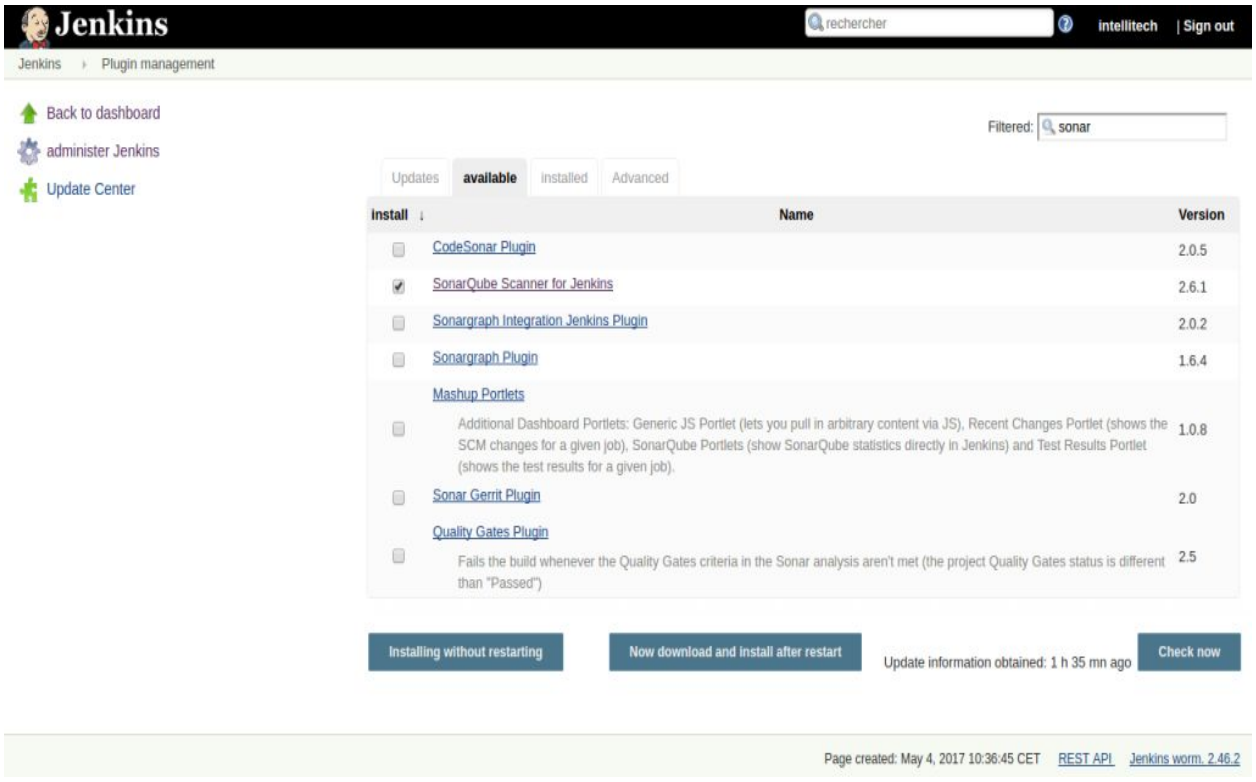
Go to **profiles** tag and add

```
<profile>
   <id>sonar</id>
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>

        <properties>

          <sonar.host.url>http://localhost:9000

        </sonar.host.url>

        </properties>

</profile>
```

SonarQube plugin in Jenkins

Install SonarQube in Jenkins

Login to Jenkins dashboard and navigate to **Manage Jenkins >> Manage Plugins >> Available Tab** and select “SonarQube Scanner for Jenkins” plugin and install.



Jenkins › Plugin management

Filtered: sonar

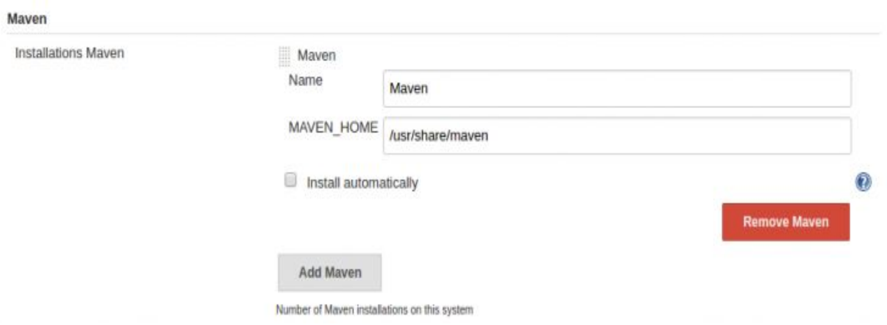
Install ↓	Name	Version
☐	CodeSonar Plugin	2.0.5
✓	SonarQube Scanner for Jenkins	2.6.1
☐	Sonargraph Integration Jenkins Plugin	2.0.2
☐	Sonargraph Plugin	1.6.4
☐	Mashup Portlets Additional Dashboard Portlets: Generic JS Portlet (lets you pull in arbitrary content via JS), Recent Changes Portlet (shows the SCM changes for a given job), SonarQube Portlets (show SonarQube statistics directly in Jenkins) and Test Results Portlet (shows the test results for a given job).	1.0.8
☐	Sonar Gerrit Plugin	2.0
☐	Quality Gates Plugin Fails the build whenever the Quality Gates criteria in the Sonar analysis aren't met (the project Quality Gates status is different than "Passed")	2.5

Update information obtained: 1 h 35 mn ago

Page created: May 4, 2017 10:36:45 CET REST API Jenkins worm. 2.46.2

Configure sonarQube with Jenkins

Add **MAVEN_HOME** in Jenkins


Maven

Installations Maven

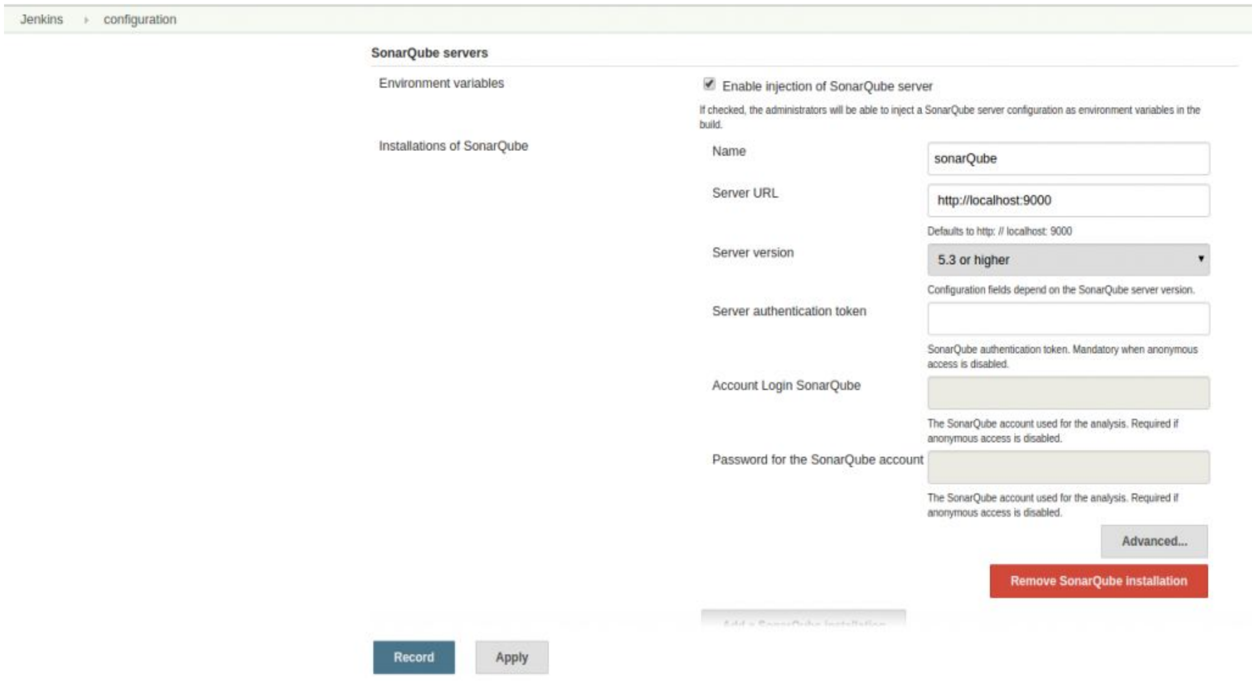
Maven

Name: Maven

MAVEN_HOME: /usr/share/maven

☐ Install automatically

Go to **Manage Jenkins >>configuring the system**, Search SonarQube servers section, Check “Enable injection of SonarQube server” and click “Add a SonarQube installation” button



SonarQube servers

Environment variables: ✓ Enable injection of SonarQube server

If checked, the administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Installations of SonarQube

Name: sonarQube

Server URL: http://localhost:9000

Defaults to http: // localhost: 9000

Server version: 5.3 or higher

Configuration fields depend on the SonarQube server version.

Server authentication token:

SonarQube authentication token. Mandatory when anonymous access is disabled.

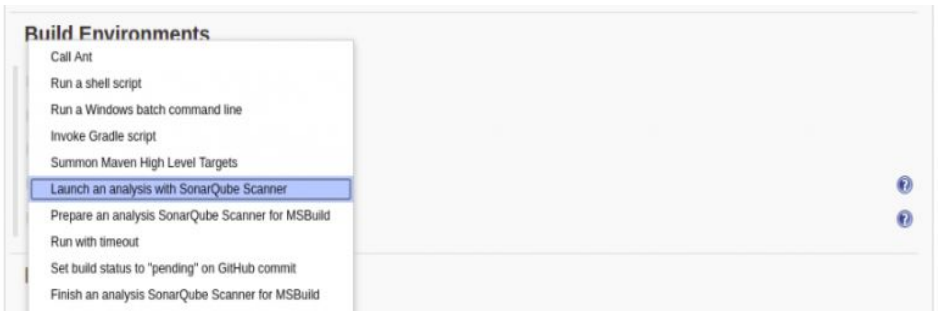
Account Login SonarQube:

The SonarQube account used for the analysis. Required if anonymous access is disabled.

Password for the SonarQube account:

The SonarQube account used for the analysis. Required if anonymous access is disabled.

In the build configuration go to **Actions following the build** section and click “Add an action after the build” button and choose “SonarQube analysis with Maven”


Build Environments
Call Ant
Run a shell script
Run a Windows batch command line
Invoke Gradle script
Summon Maven High Level Targets
Launch an analysis with SonarQube Scanner
Prepare an analysis SonarQube Scanner for MSBuild
Run with timeout
Set build status to "pending" on GitHub commit
Finish an analysis SonarQube Scanner for MSBuild

What is Jenkins? What does it do?

Jenkins is an open-source continuous integration software tool written in the Java programming language for testing and reporting on isolated changes in a larger code base in real time. The software enables developers to find and solve defects in a code base rapidly and to automate testing of their builds.

A workflow in Jenkins is called a ____ and the execution of the workflow is called a ____.

Task, Job

What's a "build trigger"?

A build trigger may be used for various purposes depending on the context of the project.

For example:

If an organization would like to have a CI/CD pipeline setup using plain Jenkins. They will have the build triggers to trigger downstream projects such as

- Integration tests
- Code scans
- Performance Tests
- End to End Tests
- Deployment.

The above stages will be chained to the parent job and can be triggered one by one or in parallel depending on the stage (this is where build trigger is used, trigger the downstream project if the parent is successful).

Build periodically can be used to run on standard jobs (if you have a team setup which has deployment to master every evening). Then you can set up the test jobs to build periodically late evenings at a fixed time (also the test job can be triggered based on success as explained in (1)

Poll SCM is nothing but checking if there is new code committed in your repository and build based on that.

What's the point of having a pipeline?

What's AMI? IAM? EC2, S3, EBS?

AMI

An Amazon Machine Image is a special type of virtual appliance that is used to create a virtual machine within the Amazon Elastic Compute Cloud. It serves as the basic unit of deployment for services delivered using EC2.

IAM

AWS Identity and Access Management (IAM) enables you to manage access to AWS services and resources securely. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources.

EC2

Amazon Elastic Compute Cloud forms a central part of Amazon.com's cloud-computing platform, Amazon Web Services, by allowing users to rent virtual computers on which to run their own computer applications.

S3

Amazon S3 or Amazon Simple Storage Service is a "simple storage service" offered by Amazon Web Services that provides object storage through a web service interface. Amazon S3 uses the same scalable storage infrastructure that Amazon.com uses to run its global e-commerce network.

EBS

Amazon Elastic Block Store (Amazon EBS) provides persistent block storage volumes for use with Amazon EC2 instances in the AWS Cloud. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability.

How is security handled in AWS?

General Best Practices

The process for securing EC2 instances involves principles that are applicable to any OS, whether running in a virtual machine or on premises:

- Least Access: Restrict server access from both the network and on the instance, install only the required OS components and applications, and leverage host-based protection software.
- Least Privilege: Define the minimum set of privileges each server needs in order to perform its function.
- Configuration Management: Create a baseline server configuration and track each server as a configuration item. Assess each server against the current recorded baseline to identify and flag any deviations. Ensure each server is configured to generate and securely store appropriate log and audit data.
- Change Management: Create processes to control changes to server configuration baselines.
- Audit Logs: Audit access and all changes to EC2 instances to verify server integrity to ensure only authorized changes are made.

What is the difference between key pairs and security groups?

A **key pair** consists of an actual key, remember the .PEM file we've stored locally and used to access our server through the web portal or via PuTTY.

AWS security groups (SGs) are associated with EC2 instances and provide security at the protocol and port access level. Each security group – working much the same way as a firewall – contains a set of rules that filter traffic coming into and out of an EC2 instance.

How does Maven integrate with Jenkins?

Maven can be installed on the remote server along with Jenkins, then from within Jenkins, we can import Maven and run it as a Jenkins Job.

INTERVIEW QUESTIONS

1. **Tell me about yourself, technologies, and projects**

I am a full-stack Java developer, I have experience using Angular, (HTML/CSS/JavaScript & TypeScript) Java with Spring, Hibernate, Servlets & REST. Currently I am working on Test Driven Development for a social media platform using Jasmine & Karma.

2. **Any previous experience before this?**

I have prior experience working with the SDLC Pipeline, manual testing, SCRUM principals, and IT, I have IT experiece from working at Amazon.

3. **Year I earned my degree**

2015

4. **What is OOP, and how does it incorporate into Java?**

Object-oriented programming is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. There are four pillars of OOP, Abstraction, Encapsulation, Polymorphism & Inheritance.

5. **How have we incorporated that into our projects?**

Virtually everything we do in Java incorporates the elements of OOP. Everytime we overload a constructor, override a class, create a subclass, use access modifiers, we are incorporating OOP.

6. **Spring MVC, how have we used it?**

The Spring Web MVC framework provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications.

The Model encapsulates the application data and in general they will consist of POJO.

The View is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.

The Controller is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.

Spring MVC is currently being used in our social media project to handle user input on the server-side.

7. **Explain Model and View?**

Simply put, the model can supply attributes used for rendering views.

To provide a view with usable data, we simply add this data to its Model object. Additionally, maps with attributes can be merged with Model instances:

In Spring MVC, view resolvers enable you to render models in a browser without tying you to a specific view technology like JSP, Velocity, XML…etc. There are two interfaces that are important to the way Spring handles views are ViewResolver and View. ... The default configuration file is /WEB-INF/views.xml.

8. **Establishing connection with JDBC?**

Import JDBC packages.
Load and register the JDBC driver.
Open a connection to the database.
Create a statement object to perform a query.
Execute the statement object and return a query resultset.
Process the resultset.
Close the resultset and statement objects.
Close the connection.

9. **Gathering information from multiple tables with JDBC?**

To hide the complexity of your data structure, you could use a combination of SQL views and unions to provide a unified view of your data over time.

The SQL Group By predicate would then allow you to aggregate your data by time slices.

Given your example, you would have a view ViewMarch defined as :

```
CREATE TABLE TableMarch1(Time timestamp, Symbol integer, value integer);
CREATE TABLE TableMarch2(Time timestamp, Symbol integer, value integer);

CREATE VIEW ViewMarch AS
   SELECT Time, Symbol, Value FROM TableMarch1
   UNION
   SELECT Time, Symbol, Value from TableMarch2;
```

You could then compute your average symbol value per hour, for example, using a query like :

```
SELECT LEFT(Time, 13) AS Period, Symbol, AVG(Value)
   FROM ViewMarch
   GROUP BY Period, Symbol;
```

But watch out for the performance cost. I don't know how MonetDB will optimize queries over unions.

10. **Hibernate vs JDBC?**

Like JDBC, Hibernate supports Structured Query Language (SQL) Hibernate Query Language (HQL) is similar to SQL in that it is an object-oriented query language. However, it doesn't operate on tables like SQL but rather uses persistent objects and their properties.

11. **UI to application?**

Lol what?

12. **Benefits of web services**

The intent behind a web service is to drive the Internet as a transactional tool rather than simply a visual tool. These application-to-application interactions are driven by, and built on, existing standards such as: Extensible Markup Language

In my own words, web services, such as AWS, are helpful in the fact that you don't need enterprise grade hardware to get started, a startup company can rent software as a service and scale based on the demands of their company at the time they are needed.

13. **Benefits of CSS**

Easier to maintain and update.
Greater consistency in design.
More formatting options.
Lightweight code.
Faster download times.
Search engine optimization benefits.
Ease of presenting different styles to different viewers.
Greater accessibility.

14. **Could we only use HTML and JS?**

Angular adds TypeScript and directives

15. **How have we used JS in our project?**

JavaScript provides front-end functionality such as our front end services plus the jasmine and karma that test them.

16. **Using JS in our client side**

Well, duh...

17. **Tomcat vs Weblogic**

Tomcat is good but weblogic is bad

18. **Using GitHub repos**

Github is good

19. **Deployment of code in EC2**

Create a keypair then launch a VM

20. **AWS**

Amzon Web sErvices.

21. **JSON? What is it, how do we use it?**

JavaScript Object Notation

JSON is a lightweight format for storing and transporting data. JSON is often used when data is sent from a server to a web page. JSON is "self-describing" and easy to understand.

22. **PL/SQL?**

PL/SQL is Oracle Corporation's procedural extension for SQL and the Oracle relational database. PL/SQL is available in Oracle Database, TimesTen in-memory database, and IBM DB2. Oracle Corporation usually extends PL/SQL functionality with each successive release of the Oracle Database.

23. **DDL, DML, DQL and TCL**

Data Definition Language (DDL) is a standard for commands that define the different structures in a database. DDL statements create, modify, and remove database objects such as tables, indexes, and users. Common DDL statements are CREATE, ALTER, and DROP.

24. **Joins in SQL**

SQL. Join (Inner, Left, Right and Full Joins) A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them.

25. **L1 vs L2 caching**

L1 is "level-1" cache memory, usually built onto the microprocessor chip itself. For example, the Intel MMX microprocessor comes with 32 thousand bytes of L1. L2 (that is, level-2) cache memory is on a separate chip (possibly on an expansion card) that can be accessed more quickly than the larger "main" memory.

26. **SonarCloud and SonarQube**

SonarCloud is a cloud service offered by SonarSource and based on SonarQube. SonarQube is a widely adopted open source platform to inspect continuously the quality of source code and detect bugs, vulnerabilities and code smells in more than 20 different languages.

27. **Exceptions vs Errors**

An Error "indicates serious problems that a reasonable application should not try to catch." An Exception "indicates conditions that a reasonable application might want to catch." Error along with RuntimeException & their subclasses are unchecked exceptions. All other Exception classes are checked exceptions.

28. **StringBuilder vs StringBuffer → why would use one or the other**

String is immutable whereas StringBuffer and StringBuilder are mutable classes. StringBuffer is thread safe and synchronized whereas StringBuilder is not, that's why StringBuilder is more faster than StringBuffer. String concat + operator internally uses StringBuffer or StringBuilder class.

29. **How do you perform JUnit on servlets?**

You can do this using Mockito to have the mock return the correct params, verify they were indeed called (optionally specify number of times), write the 'result' and verify it's correct.

30. **Thread vs. Runnable**

The significant differences between extending Thread class and implementing Runnable interface: ... When we extend Thread class, each of our thread creates unique object and associate with it. When we implements Runnable, it shares the same object to multiple threads.

31. **Start from a list of employees and create a new list without duplicates**

Create an auxiliary array temp[] to store unique elements.
Traverse input array and one by one copy unique elements of arr[] to temp[]. Also keep track of count of unique elements. Let this count be j.
Copy j elements from temp[] to arr[] and return j

```
// simple java program to remove

// duplicates



class Main

{

    // Function to remove duplicate elements

    // This function returns new size of modified

    // array.

    static int removeDuplicates(int arr[], int n)

    {

        // Return, if array is empty

        // or contains a single element
        if (n==0 || n==1)
            return n;

        int[] temp = new int[n];

        // Start traversing elements
        int j = 0;
        for (int i=0; i<n-1; i++)
            // If current element is not equal
            // to next element then store that
            // current element
            if (arr[i] != arr[i+1])
                temp[j++] = arr[i];

        // Store the last element as whether
        // it is unique or repeated, it hasn't
        // stored previously
        temp[j++] = arr[n-1];

        // Modify original array
        for (int i=0; i<j; i++)
            arr[i] = temp[i];

        return j;
    }

    public static void main (String[] args)
    {
        int arr[] = {1, 2, 2, 3, 4, 4, 4, 5, 5};
        int n = arr.length;
        n = removeDuplicates(arr, n);


        // Print updated array

        for (int i=0; i<n; i++)

           System.out.print(arr[i]+" ");

    }

}
```

32. **What are single page applications?**

A single-page application is a web application or web site that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server

33. **How do you create a custom unchecked exception?**

Yes - derive it from RuntimeException

34. **What is the Garbage collector in Java and what does it do?**

The garbage collector is a program which runs on the Java Virtual Machine which gets rid of objects which are not being used by a Java application anymore. It is a form of automatic memory management. In the above code, the String s is being created on each iteration of the for loop.

35. **What is a path space variable and path space value?**

A path space is a space in a URL path, to use one we put a $

36. **You know what a constructor is, what is a destructor and what does it do?**

It’s what you’d use for garbage collection in a .net language such as C that doesn’t automatically handle garbage collection by default.

37. **What is the difference between a hash map and a hash table?**

There are several differences between HashMap and Hashtable in Java: Hashtable is synchronized, whereas HashMap is not. This makes HashMap better for non-threaded applications, as unsynchronized Objects typically perform better than synchronized ones. Hashtable does not allow null keys or values.

38. **What is the difference between .equal and ==?**

In general both equals() and "==" operator in Java are used to compare objects to check equality but here are some of the differences between the two: Main difference between .equals() method and == operator is that one is method and other is operator.

39. **What is dynamic polymorphism and static polymorphism?**

In simple terms : Static polymorphism : Same method name is overloaded with different type or number of parameters in same class (different signature). Targeted method call is resolved at compile time. Dynamic polymorphism: Same method is overridden with same signature in different classes.

40. **How do you skip the Finally statement in a try/catch block?**

If there is no exception occurred in the code which is present in try block then first, the try block gets executed completely and then control gets transferred to finally block (skipping catch blocks). If a return statement is encountered either in try or catch block.

41. **How to reverse a string in Java? without the reverse() method in StringBuilder.**

```
// Java program to Reverse a String  by
// converting string to characters  one
// by one
import java.lang.*;
import java.io.*;
import java.util.*;

// Class of ReverseString
class ReverseString
{
   public static void main(String[] args)
   {
      String input = "Nate";

      // convert String to character array
      // by using toCharArray
      char[] try1 = input.toCharArray();

      for (int i = try1.length-1; i>=0; i--)
         System.out.print(try1[i]);
   }
}
Output:

etaN
```

How to implement a hashcode function for a class.

```
@Override
public int hashCode() {
   final int prime = 31;
   int result = 1;
   result = prime * result + ((email == null) ? 0 : email.hashCode());
   result = prime * result + (int) (id ^ (id >>> 32));
   result = prime * result + ((name == null) ? 0 : name.hashCode());
   return result;
}
```

42. What is a daemon thread?

A daemon thread is a thread that does not prevent the JVM from exiting when the program finishes but the thread is still running. An example for a daemon thread is the garbage collection. You can use the setDaemon(boolean) method to change the Thread daemon properties before the thread starts.

43. How do you serialize an object?

To serialize an object means to convert its state to a byte stream so that the byte stream can be reverted back into a copy of the object. A Java object is serializable if its class or any of its superclasses implements either the java.io.Serializable interface or its subinterface, java.io.Externalizable.

44. You have a table of employees with country and state as columns, so how can you make a query to return the number of employees in each country and state?

The SQL COUNT() function returns the number of rows in a table satisfying the criteria specified in the WHERE clause. It sets the number of rows or non NULL column values. COUNT() returns 0 if there were no matching rows.

45. What is Rest and Rest assured and how do we use it?

Rest Assured Tutorial for REST API Automation Testing. This is a series of Rest Assured Tutorial which is one of the most used library for REST API Automation Testing. Rest-Assured is a Java based library that is used to test RESTful Web Services. This library behaves like a headless Client to access REST web services.

46. What do you know about the agile scrum methodology and how have you used it?

We used this during project two. Burn down charts and scrum meetings were used as well as daily standups to check on the progress of each team's project.

47. What is a Web container?

A web container is the component of a web server that interacts with Java servlets. A web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access-rights.

48. **Can you discuss Mockito and JUnit testing?**

Mockito is an open source testing framework for Java released under the MIT License. The framework allows the creation of test double objects in automated unit tests for the purpose of test-driven development or behavior-driven development. The framework's name and logo are a play on mojitos, a type of drink.

49. **What is Hibernate and why do we use it?**

Hibernate ORM is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database.

50. **What is the MVC design pattern?**

Model view controller

51. **Can you discuss the response life cycle from front to back?**

Flow of Data through Krowd

Client - > Servlet - >service -> DAO - > DB -> DAO -> Service -> Servlet -> Client

Client sends a login request, the HTTP get method is sent to the servlet, which passes it to the service which is running methods checking if the username and password are valid. From there, the data is handed to the DAO which checks to make sure the user ID corresponds

52. **How did you setup your web services?**

Amazon

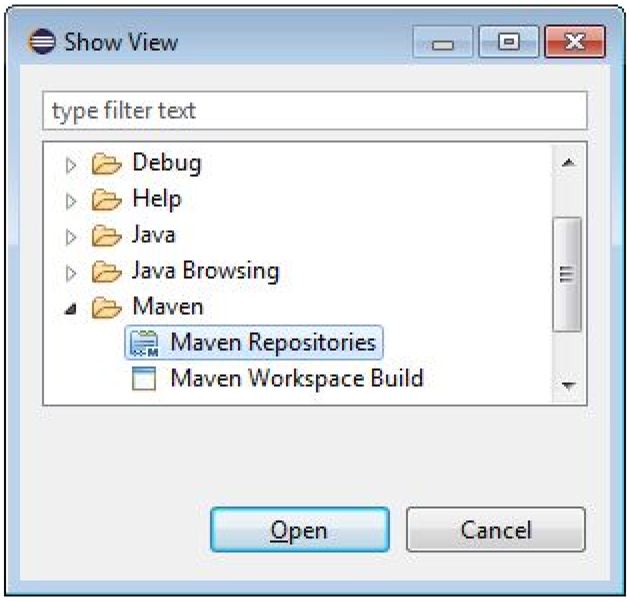
53. **What do you know about Jenkins, the S3 bucket, RDS, Travis CI and sonarcloud?**

We hosted jenkins on our AWS server S3 bucket is for storing static files, RDS is for hosting a database, travis CI is for testing git hub files, and sonarcloud is for testing code smells.

Jake Mulrenin Interview Questions

How do you add dependencies in Maven without using the POM?

1. On the top menu bar, open Window -> Show View -> Other
2. In the Show View window, open Maven -> Maven Repositories

3. 

4. In the window that appears, right-click on Global Repositories and select Go Into
5. Right-click on "central (http://repo.maven.apache.org/maven2)" and select "Rebuild Index"
    - Note that it will take a while to complete the download
6. Once indexing is complete, Right-click on the project -> Maven -> Add Dependency and start typing the name of the project you want to import (such as "hibernate").
    - The search results will auto-fill in the "Search Results" box below.

Given a table of Names, States, and Countries, write a query statement that returns the number of entries for each state and each country.

im assuming you'd GroupBy state and GroupBy country then count() the results

What is CSS?

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

What are Servlets?

A Java servlet is a Java software component that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement web containers for hosting web applications on web servers and thus qualify as a server-side servlet web API.

Write a unit test for a servlet.

```
import static org.junit.Assert.*;
import static org.mockito.Mockito.*;
import java.io.*;
import javax.servlet.http.*;
import org.apache.commons.io.FileUtils;
import org.junit.Test;

public class TestMyServlet extends Mockito{

   @Test
   public void testServlet() throws Exception {
      HttpServletRequest request = mock(HttpServletRequest.class);
      HttpServletResponse response = mock(HttpServletResponse.class);

      when(request.getParameter("username")).thenReturn("me");
      when(request.getParameter("password")).thenReturn("secret");

      StringWriter stringWriter = new StringWriter();
      PrintWriter writer = new PrintWriter(stringWriter);
      when(response.getWriter()).thenReturn(writer);

      new MyServlet().doPost(request, response);

      verify(request, atLeast(1)).getParameter("username"); // only if you want to verify
username was called...
      writer.flush(); // it may not have been flushed yet...
      assertTrue(stringWriter.toString().contains("My expected string"));
   }
}
```

Write a recurrent function for Fibonacci sequence.

```
public int fibonacci(int n)  {
    if(n == 0)
        return 0;
    else if(n == 1)
      return 1;
   else
      return fibonacci(n - 1) + fibonacci(n - 2);
}
```

How do you deploy and application through weblogic?

Install DataDirect Connect for JDBC.
Create a connection pool using the WebLogic Server Administration Console
Create a data source using the Administration Console.
Generate the JSP test page. A sample JSP test page is provided in the WebLogicTest.war file, which is available with this document.
Deploy the WebLogic web application to the WebLogic server.
Run the WebLogic web application.

Write the hash code function for an employee with a name, age, and salary.

What is a SPA?

A single-page application is a web application or web site that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server.

Thread vs Runnable

The Callable interface is similar to Runnable, in that both are designed for classes whose instances are potentially executed by another thread. A Runnable, however, does not return a result and cannot throw a checked exception.

String Builder vs String Buffer

String is immutable, if you try to alter their values, another object gets created, whereas StringBuffer and StringBuilder are mutable so they can change their values.
Thread-Safety Difference: The difference between StringBuffer and StringBuilder is that StringBuffer is thread-safe.

What do we use hash codes for?

hashCode() is used for bucketing in Hash implementations like HashMap , HashTable , HashSet , etc. The value received from hashCode() is used as the bucket number for storing elements of the set/map.

What is a Controller?

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns. Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.

What is a final class?

Writing Final Classes and Methods. You can declare some or all of a class's methods final. You use the final keyword in a method declaration to indicate that the method cannot be overridden by subclasses. The Object class does this—a number of its methods are final . ... A class that is declared final cannot be subclassed.

Write a function that would delete duplicates from a list of employees and return that list.

Using Iterator
Approach:

Get the ArrayList with duplicate values.
Create another ArrayList.
Traverse through the first arraylist and store the first appearance of each element into the second arraylist using contains() method.

The second ArrayList contains the elements with duplicates removed.
Below is the implementation of the above approach:

```
// Java program to remove duplicates from ArrayList


import java.util.*;


public class GFG {

    // Function to remove duplicates from an ArrayList

    public static <T> ArrayList<T> removeDuplicates(ArrayList<T> list)

    {

        // Create a new ArrayList

        ArrayList<T> newList = new ArrayList<T>();


        // Traverse through the first list

        for (T element : list) {


            // If this element is not present in newList

            // then add it

            if (!newList.contains(element)) {

                newList.add(element);

            }

        }


        // return the new list

        return newList;

    }


    // Driver code

    public static void main(String args[])

    {
        // Get the ArrayList with duplicate values

        ArrayList<Integer>

            list = new ArrayList<>(

                Arrays

                    .asList(1, 10, 1, 2, 2, 3, 3, 10, 3, 4, 5, 5));

        // Print the Arraylist

        System.out.println("ArrayList with duplicates: "

                           + list);

        // Remove duplicates

        ArrayList<Integer>

            newList = removeDuplicates(list);

        // Print the ArrayList with duplicates removed

        System.out.println("ArrayList with duplicates removed: "

                           + newList);

    }

}
```

```
ArrayList with duplicates: [1, 10, 1, 2, 2, 3, 3, 10, 3, 4, 5, 5]
ArrayList with duplicates removed: [1, 10, 2, 3, 4, 5]
```

What is deadlock?

. Deadlock describes a situation where two or more threads are blocked forever, waiting for each other. Deadlock occurs when multiple threads need the same locks but obtain them in different order.

Why do we write Services in Java?

Generally the DAO is as light as possible and exists solely to provide a connection to the DB, sometimes abstracted so different DB backends can be used.

The service layer is there to provide logic to operate on the data sent to and from the

DAO and the client. Very often these 2 pieces will be bundled together into the same module, and occasionally into the same code, but you'll still see them as distinct logical entities.

Another reason is security - If you provide a service layer that has no relation to the DB, then is it more difficult to gain access to the DB from the client except through the service. If the DB cannot be accessed directly from the client (and there is no trivial DAO module acting as the service) then all an attacker who has taken over the client can do is attempt to hack the service layer as well before he gets all but the most sanitised access to your data.

What is JSON?

JSON stands for JavaScript Object Notation

JSON is a lightweight format for storing and transporting data

JSON is often used when data is sent from a server to a web page

JSON is "self-describing" and easy to understand

What are Connections used for in Java?

When a Java application needs a database connection, one of the DriverManager.getConnection() methods is used to create a JDBC connection. The URL used is dependent upon the particular database and JDBC driver. It will always begin with the "jdbc:" protocol, but the rest is up to the particular vendor.

What is an interface?

An interface in the Java programming language is an abstract type that is used to specify a behavior that classes must implement. They are similar to protocols. Interfaces are declared using the interface keyword, and may only contain method signature and constant declarations.

Write a function that returns the number of times a Servlet has been invoked.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class SiteHitCounter implements Filter {

   private int hitCount;

   public void  init(FilterConfig config) throws ServletException {
      // Reset hit counter.
      hitCount = 0;
   }

   public void  doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
      throws java.io.IOException, ServletException {

      // increase counter by one
      hitCount++;

      // Print the counter.
      System.out.println("Site visits count :"+ hitCount );

      // Pass request back down the filter chain
      chain.doFilter(request,response);
   }

   public void destroy() {
      // This is optional step but if you like you
      // can write hitCount value in your database.
   }
}
```

What is the standard development life cycle?

1. Requirement Analysis. The requirements of the software are determined at this stage. ...

2. Design. Here, the software and system design is developed according to the instructions provided in the 'Requirement Specification' document. ...

3. Implementation & Coding. ...

4. Testing. ...

5. Maintenance.